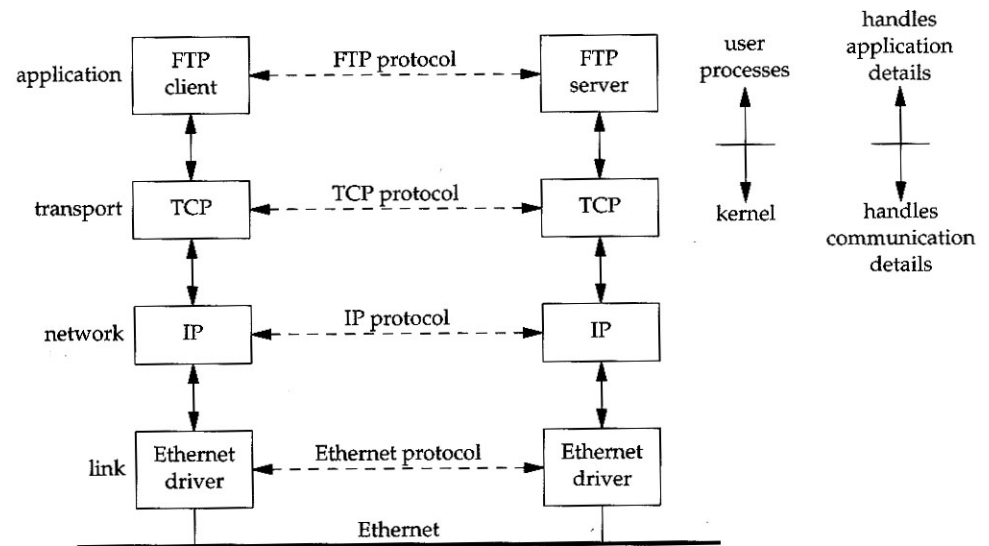


Interligação de Redes
Redes IP

FEUP/DEEC
Redes de Computadores
MIEIC – 2009/10
José Ruela

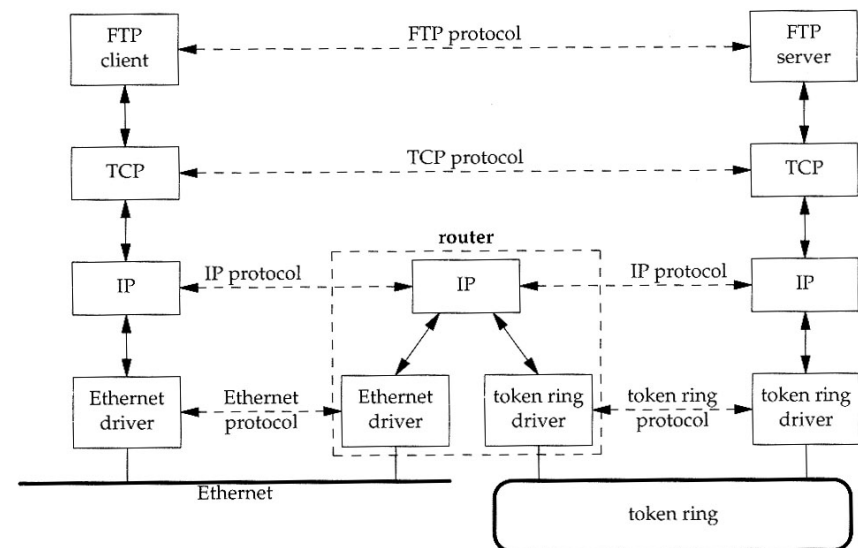
Pilha protocolar TCP/IP

- Acesso à rede (física)
 - » Interface ao meio físico
 - Carta de rede, *device driver*
- Rede (*internetworking*)
 - » Transferência de pacotes na rede (através de várias redes físicas)
 - Protocolos de encaminhamento (e.g., RIP, OSPF, BGP)
 - » IP – *Internet Protocol*
 - » ICMP – *Internet Control Message Protocol*
 - » IGMP – *Internet Group Management Protocol*
- Transporte
 - » Multiplexagem de fluxos de dados entre máquinas (*hosts*)
 - TCP – *Transmission Control Protocol*
 - UDP – *User Datagram Protocol*
- Aplicação
 - » Serviços de utilizador
 - Telnet, FTP, SMTP, HTTP, etc.
 - » Modelo tradicional: cliente-servidor



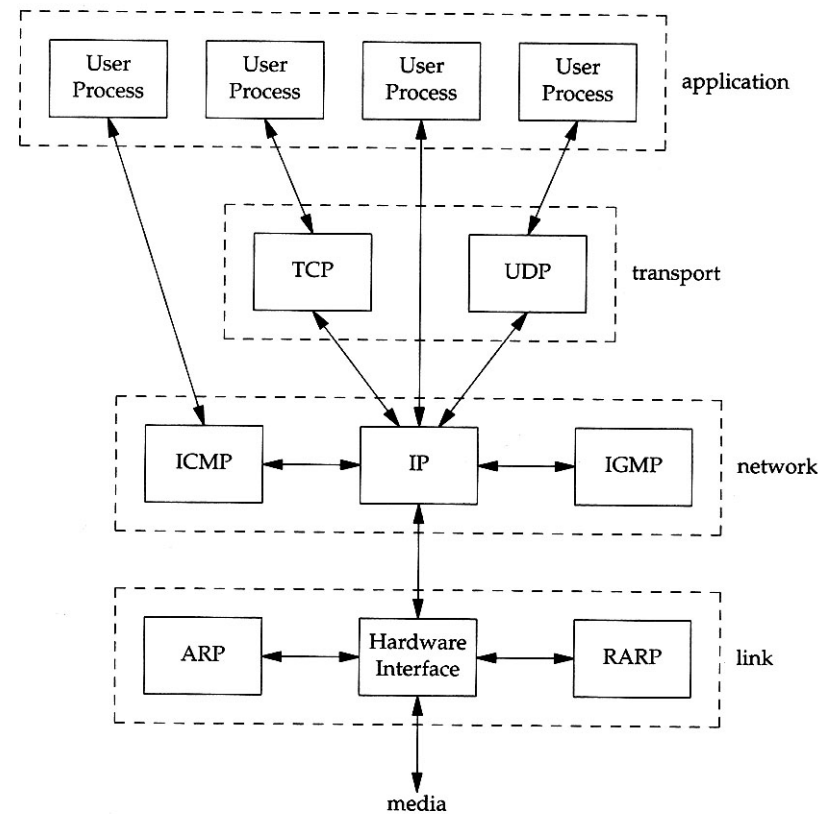
Interligação de redes

- Redes de diferentes organizações e usando diferentes tecnologias de acesso devem poder comunicar
- Internet – rede única, virtual
 - » *End-Systems* – computadores (*hosts*)
 - » *Intermediate-Systems* – *routers*
 - » Comunicação global
 - » Interligação de redes com *routers*
 - Um *router* tem uma interface por cada rede física que interliga
- Comunicação protocolar
 - » Camadas de Aplicação e Transporte
 - Extremo-a-extremo (entre *hosts*)
 - » Camada de Rede
 - Entre máquinas adjacentes no nível IP

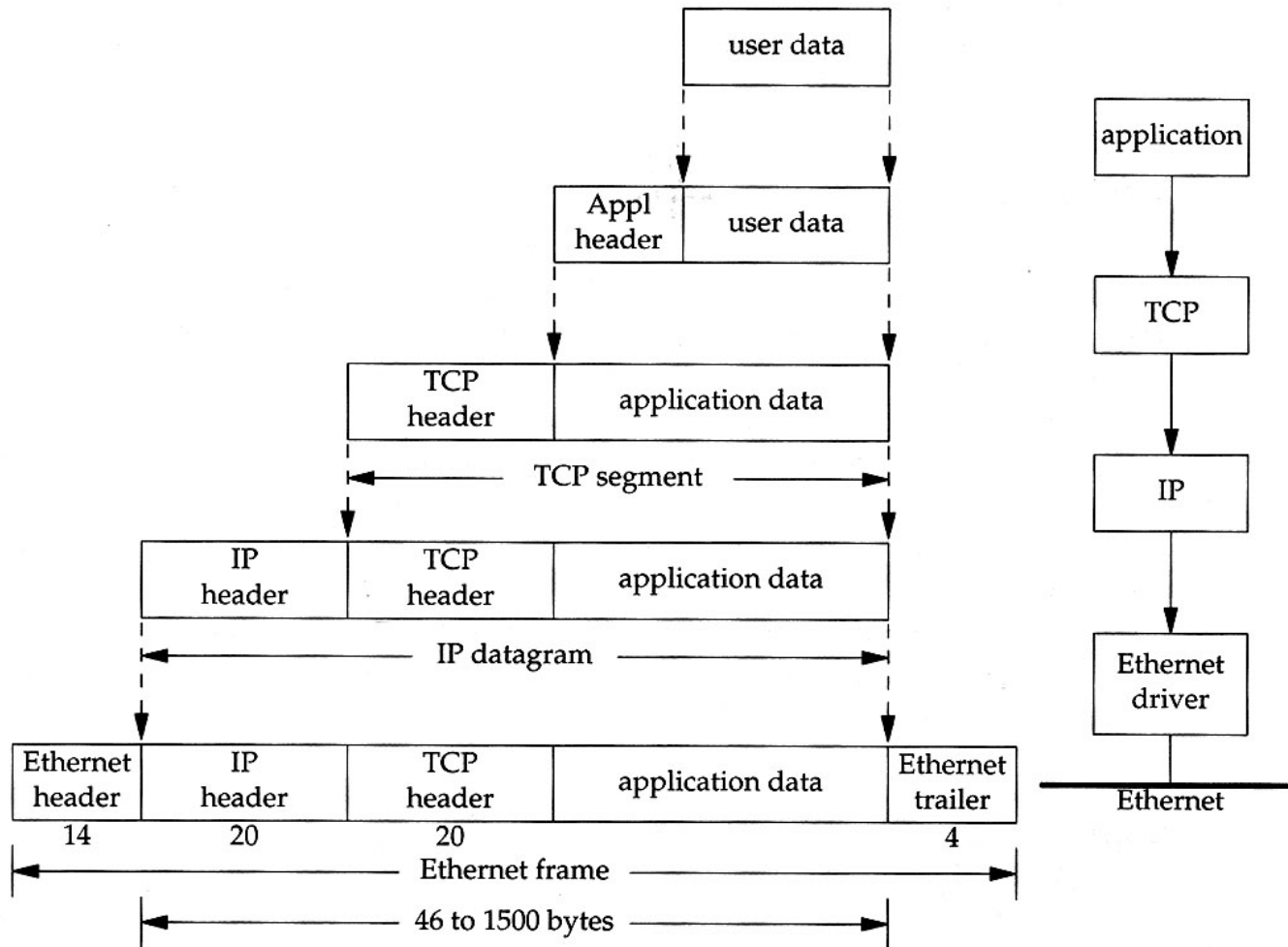


Arquitetura TCP/IP

- TCP
 - » Serviço orientado à conexão
 - » Transporte fiável entre máquinas
- UDP
 - » Serviço sem conexão (*connectionless*)
 - » Transporte não fiável
- IP
 - » Protocolo central da pilha
 - » Encaminha datagramas (*connectionless*)
- ICMP
 - » Auxiliar do IP
 - » Envia mensagens de erro
- IGMP
 - » Gere grupos *multicast*
- ARP, RARP
 - » Resolução de endereços IP em endereços físicos (por exemplo, endereços MAC)



Encapsulamento



Encapsulamento Ethernet e IEEE 802

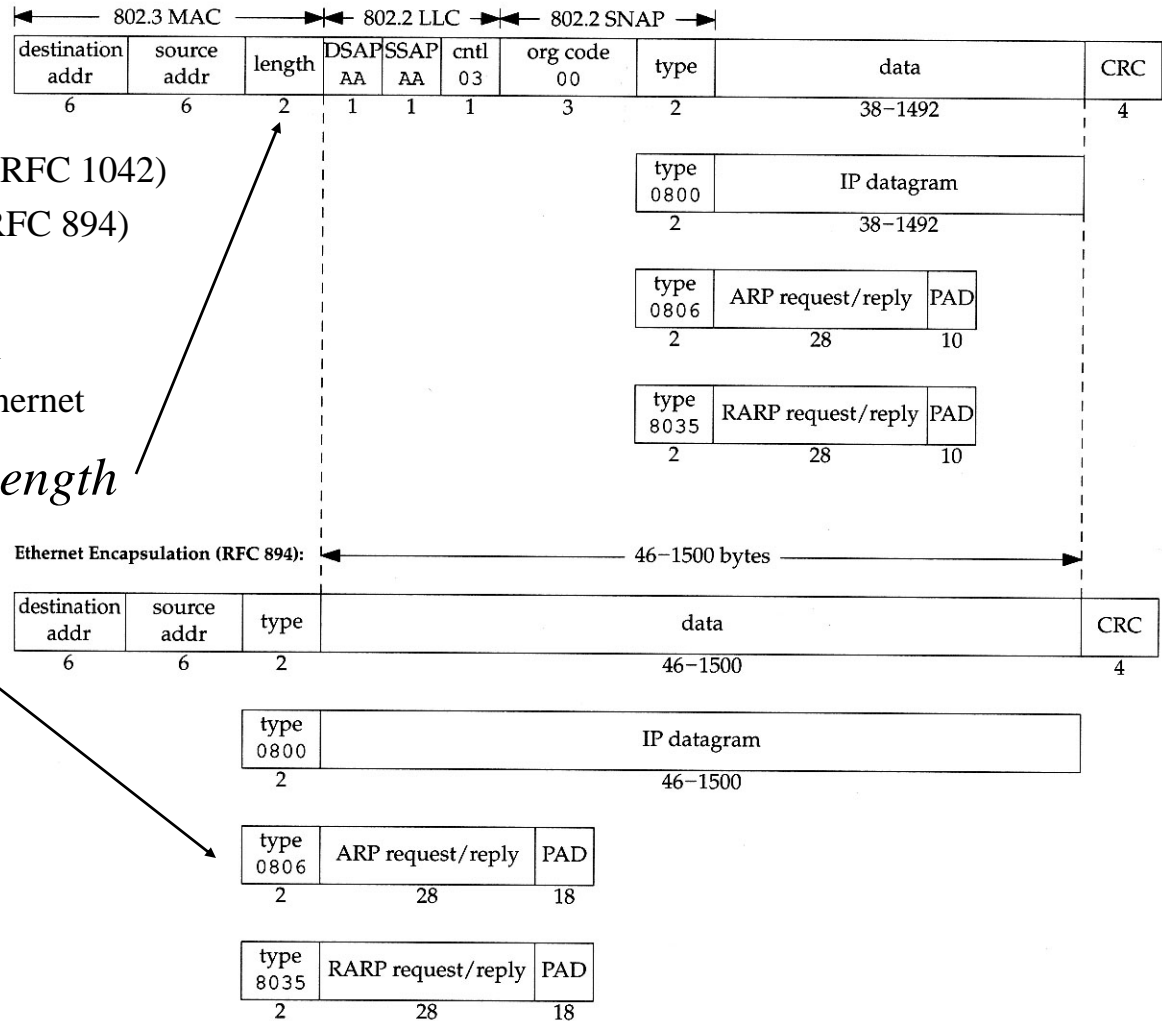
Cartas Ethernet

- » Devem permitir
 - Encapsulamento IEEE 802 (RFC 1042)
 - Encapsulamento Ethernet (RFC 894)
- » Se suportarem os 2 tipos
 - O tipo deve ser configurável
 - *Default*: encapsulamento Ethernet

Valores válidos IEEE 802 *length*

- » Diferentes de *type* válidos

- Ex. 0x0800 = 2048



Encapsulamento baseado em LLC

- Encapsulamento de protocolos “encaminháveis” (*routed ISO protocols*)
 - Identificado por DSAP = SSAP = 0xFE
 - O primeiro octeto do campo de Dados é NLPID (*Network Layer Protocol Identifier*), administrado por ISO / ITU
 - NLPID é também usado em encapsulamento não baseado em LLC
 - Valores de NLPID
 - 0x00 *Null Network Layer / Inactive Set*
 - 0x08 ITU-T Q.933
 - 0x80 SNAP (*Subnetwork Access Protocol*)
 - Usado em encapsulamento não baseado em LLC quando o protocolo não tem NLPID associado
 - LLC suporta encapsulamento LLC/SNAP
 - 0x81 ISO CLNP
 - 0x82 ISO ES-IS
 - 0x83 ISO IS-IS
 - 0xCC IP
 - IP não é protocolo ISO mas tem NLPID associado
 - IP é normalmente encapsulado com base em LLC/SNAP (LANs, IP sobre ATM, LANE)
- Encapsulamento LLC/SNAP
 - Identificado por DSAP = SSAP = 0xAA

Encapsulamento LLC/SNAP

- O campo SNAP é constituído por cinco octetos
 - OUI *Organizationally Unique Identifier* (3 octetos)
 - PID *Protocol Identifier*, normalmente designado *Ether Type* (2 octetos)
 - Tipos de encapsulamento
 - *Routed non ISO PDUs* OUI = 0x000000
 - *Bridged IEEE 802 PDUs* OUI = 0x0080C2
-
- | | |
|---|---|
| <ul style="list-style-type: none"> • <i>PID – Routed non ISO PDUs</i> <ul style="list-style-type: none"> – 0x0800 IP (RFC 1042) – 0x0806 ARP – 0x0807 XNS – 0x6003 DECnet – 0x8035 RARP – 0x809B AppleTalk – 0x8137 IPX | <ul style="list-style-type: none"> • <i>PID – Bridged IEEE 802 PDUs</i> <ul style="list-style-type: none"> – 0x0001/0007 IEEE 802.3 – 0x0002/0008 IEEE 802.4 – 0x0003/0009 IEEE 802.5 – 0x0004/000A FDDI – 0x000E BPDUs |
|---|---|

Desmultiplexagem

- Cabeçalho TCP/UDP (porta)

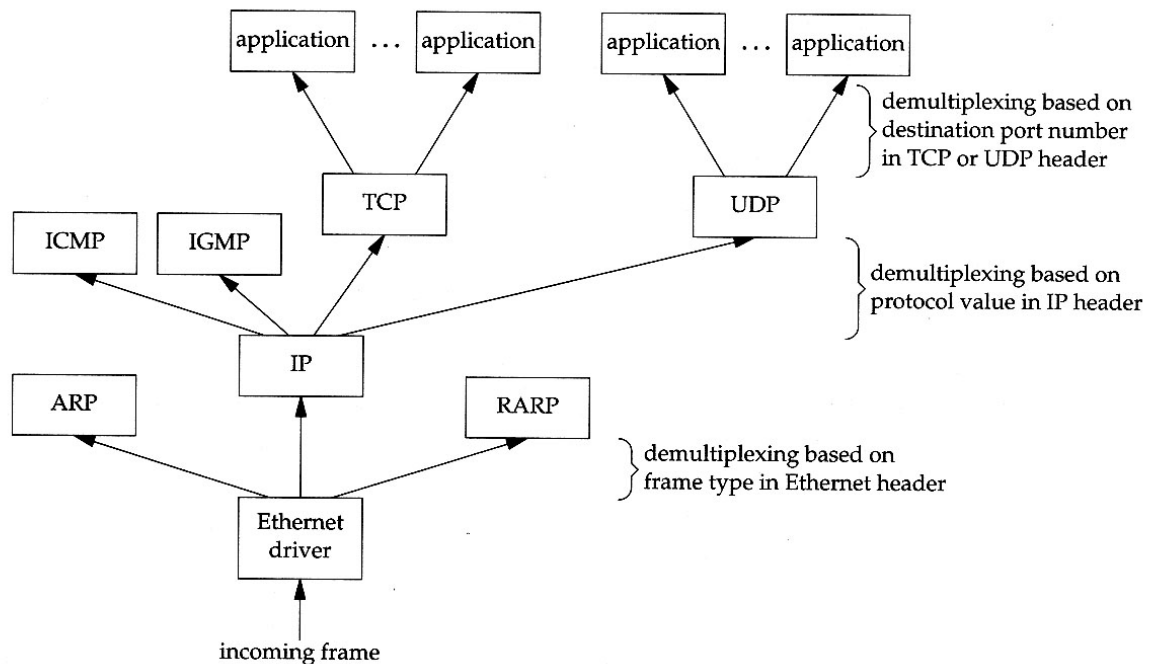
- FTP → 21
- Telnet → 23
- HTTP → 80
- SMTP → 25

- Cabeçalho IP (protocolo)

- ICMP → 1
- IGMP → 2
- TCP → 6
- UDP → 17

- Cabeçalho Ethernet (tipo)

- IP → 0x0800
- ARP → 0x0806
- RARP → 0x8035
- IPX → 0x8037
- IPv6 → 0x86DD
- MPLS → 0x8847



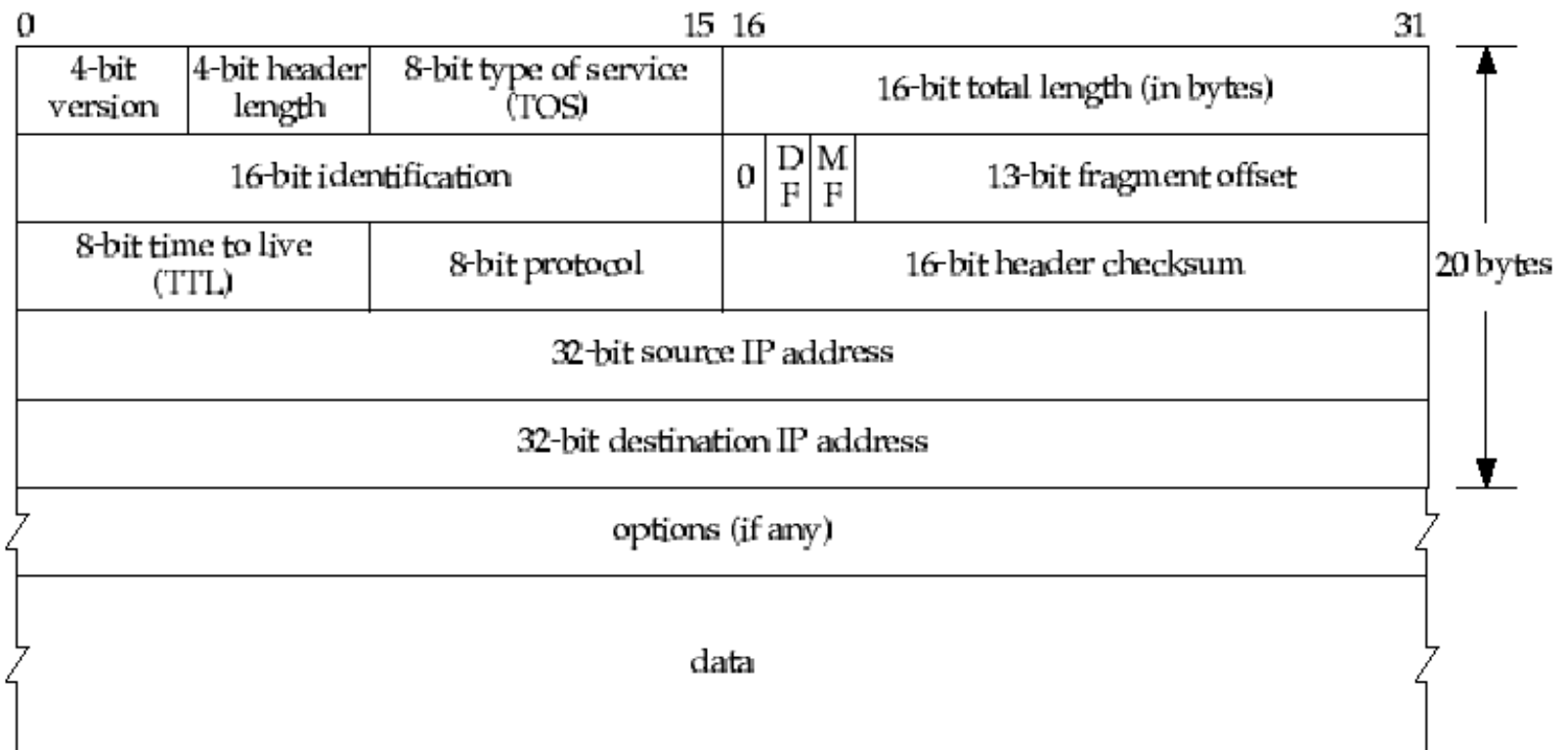
Internet Protocol

- IP
 - » Comutação de datagramas
 - » Serviço não fiável / *best effort*
 - Entrega no destino não garantida (rede pode perder / descartar datagramas)
 - Tempo máximo de entrega não garantido
 - » Datagramas para um mesmo destino podem seguir caminhos diferentes e portanto ser entregues fora de ordem

- Cada *router*
 - » Analisa o endereço de destino presente no cabeçalho de cada datagrama
 - » Comuta o datagrama recebido para uma das suas interfaces
 - » Em situações de erro
 - Elimina datagrama
 - Envia mensagem de erro ICMP para a origem

Cabeçalho IP

IP Header



Cabeçalho IP (IPv4)

- » Versão – 4 (IPv4)
- » Comprimento do cabeçalho
 - Número de palavras (de 32 bits) do cabeçalho
 - Tamanho máximo do cabeçalho – 60 octetos
- » TOS (*type of service*)
 - O RFC 791 definiu 3 bits de precedência e 3 bits para exprimir a importância relativa entre *delay* (D), *throughput* (T) e *reliability* (R), tendo o RFC 1349 proposto usar mais um bit relacionado com o custo (C)
 - Actualmente 6 bits são usados em *DiffServ* (DS *codepoint*) e os 2 restantes estão previstos para *Explicit Congestion Notification*
- » Comprimento total
 - Comprimento total do datagrama
 - Tamanho máximo do datagrama: 65535 octetos
- » Identificação
 - Identifica univocamente um datagrama
 - Incrementado por cada datagrama enviado
- » *Flags*
 - DF – *Don't Fragment*
 - MF – *More Fragments*
- » *Fragment offset*
 - Indica a posição dum fragmento em relação ao datagrama inicial
- » TTL (*time to live*) – tempo de vida
 - Número máximo de *routers* visitáveis por um datagrama
 - Valor inicial fixado pelo *host* (e.g., 16, 32, 64)
 - Decrementado por cada *router* visitado
- » Protocolo
 - Usado para desmultiplexagem
- » *Checksum*
 - Protecção do cabeçalho (detecção de erros)
- » Endereços de origem (32 bits) e destino (32 bits)
 - Endereços IP dos *hosts* de destino e origem
- » Opções
 - Registo de rota
 - Registo de tempos
 - Encaminhamento definido pela origem
 - ◆ *Loose source routing*
 - ◆ *Strict source routing*

MTU – Maximum Transmission Unit

- Cada rede limita o tamanho do campo de dados nas respectivas tramas – este limite designa-se *Maximum Transmission Unit* (MTU)
- Se o tamanho de um datagrama IP for superior ao MTU de uma rede no seu percurso, é necessário fragmentá-lo em datagramas mais pequenos, podendo ser necessário fragmentar novamente para atravessar outra(s) rede(s)
- Se um datagrama atravessar várias redes, o menor MTU condiciona o tamanho dos datagramas IP (fragmentos) entregues no destino

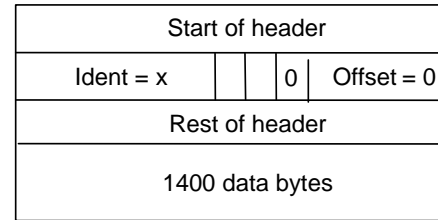
Network	MTU (bytes)
16 Mbits/sec token ring (IBM)	17914
4 Mbits/sec token ring (802.5)	4464
FDDI	4352
Ethernet	1500
IEEE 802.3/802.2	1492
X.25	576
Point-to-Point (low delay)	296

Fragmentação de pacotes IP

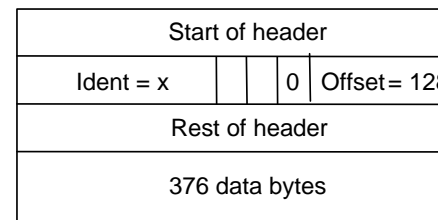
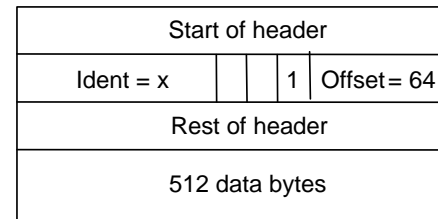
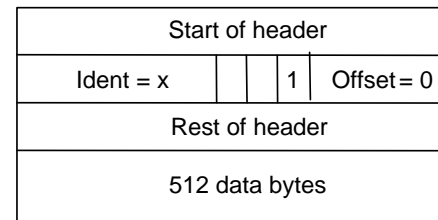
- A fragmentação é realizada quando necessário em *routers* (a fragmentação na origem é evitada)
- A reconstrução de datagramas é feita no destino
- Informação usada na fragmentação (cabeçalho IP)
 - » *Identification*
 - Valor único atribuído a cada datagrama pelo emissor
 - Copiado em cada fragmento do datagrama original
 - » *Data length*
 - Comprimento dos dados (do fragmento) em octetos
 - » *Offset*
 - Posição do fragmento no datagrama ($n * 64$ bits)
 - » *More Fragments (MF flag)*
 - Indica que o fragmento não é o último
- Problema
 - » Perda de fragmentos
- Solução
 - » Limitar o tempo máximo de reconstrução e o tempo máximo de vida (TTL) de um datagrama

Fragmentação de pacotes IP – exemplo

Rede 1
MTU = 1500



Rede 2
MTU = 532



Valor do campo *offset* = *offset* real / 8

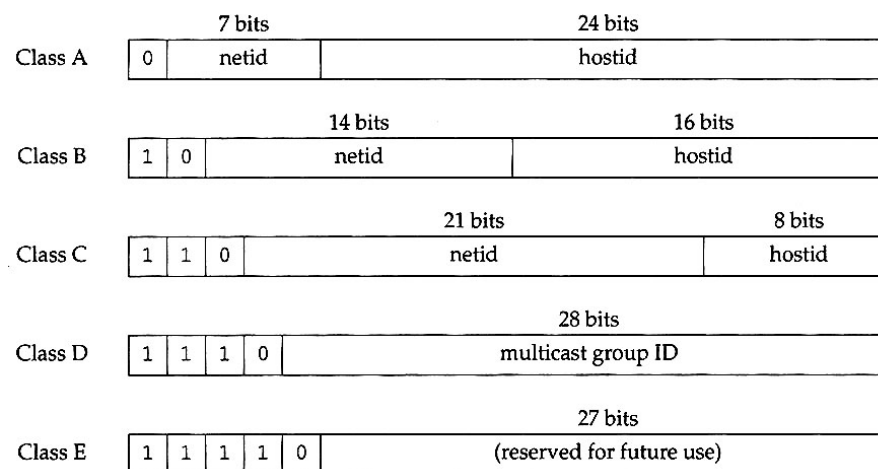
Com exceção do último fragmento, o tamanho do campo de dados deve ser múltiplo de 8 octetos

Endereçamento IP

- Um endereço IP identifica uma interface de rede (um endereço por interface) e é estruturado em duas partes
 - » *netid* – identificador da rede
 - » *hostid* – identificador da máquina (*host, router*)
- Um endereço IPv4 é constituído por 32 bits
 - » Exemplo: 142.252.13.33 (*dotted decimal notation*)
- Atribuição de endereços
 - » A gestão e coordenação de endereços IP é da responsabilidade do ICANN (*Internet Corporation for Assigned Names and Numbers*)
 - O ICANN delega partes do espaço de endereçamento a outras autoridades, que por sua vez atribuem endereços a fornecedores de serviços IP (ISP) e outras empresas
 - » A atribuição de *hostid* é da responsabilidade do Administrador de cada rede
 - *All 0s*: representa a subrede identificada por *<netid>*
 - *All 1s*: endereço de *broadcast* na subrede identificada por *<netid>*
- O esquema de endereçamento inicialmente usado na Internet era baseado em classes (*classful addressing*)
 - » Neste esquema a demarcação dos campos *netid* e *hostid* é feita ao nível de octeto (24, 16 ou 8 bits usados para atribuição de *hostid*)

Classful addressing – classes de endereços

- Classe A
 - » 126 redes: *netid* = 1 a 126
 - *netid* = all 0s: reservado
 - *netid* = 127: interface de *loopback*
 - *netid* = 10: endereços privados
- Classe B
 - » *netid* = 128.0 a 191.255
 - » *netid* = 172.16 a 172.31: endereços privados
- Classe C
 - » *netid* = 192.0.0 a 223.255.255
 - » *netid* = 192.168.0 a 192.68.255: endereços privados
- Classe D – endereços *multicast*
- Classe E – gama reservada para uso futuro



<i>Classe</i>	<i>Valores</i>
A	0.0.0.0 → 127.255.255.255
B	128.0.0.0 → 191.255.255.255
C	192.0.0.0 → 223.255.255.255
D	224.0.0.0 → 239.255.255.255
E	240.0.0.0 → 247.255.255.255

CIDR – Classless Inter-Domain Routing

- O esquema *classful addressing* apresenta limitações graves
 - » A rigidez na atribuição de endereços contribui para o seu desperdício e portanto para um mais rápido esgotamento do espaço disponível
 - » Um problema relacionado era o aumento do número de rotas que tinham de ser mantidas pelos *routers* à medida que crescia o número de redes
- Estes problemas foram resolvidos com duas medidas
 - » Reestruturação da forma de atribuir endereços (CIDR – *Classless Inter-Domain Routing*), de modo a melhorar a eficiência do processo
 - » Agregação hierárquica de rotas, com o objectivo de reduzir o tamanho das tabelas de encaminhamento
- No esquema CIDR, o identificador (prefixo) de uma rede (*netid*) pode ter um número qualquer de bits dentro da gama possível (actualmente entre 13 e 27 bits), pelo que se torna necessário associar uma máscara ao endereço
 - » O endereço é representado na notação habitual seguida de */n*, sendo *n* o número de bits do prefixo comum a todas as máquinas da rede
 - » Com este esquema adoptou-se uma atribuição hierárquica de endereços
 - » A utilização de uma máscara é igualmente necessária no processo de constituição de subredes (*subnetting*) dentro de uma organização

Endereços especiais

Endereço	<i>netid</i>	<i>hostid</i>	<i>Source / Destination</i>
Endereço de uma rede	específico	<i>all 0s</i>	Não aplicável
<i>Direct broadcast address</i>	específico	<i>all 1s</i>	<i>Destination</i>
<i>Limited broadcast address</i>	<i>all 1s</i>	<i>all 1s</i>	<i>Destination</i>
Este <i>host</i> nesta rede	<i>all 0s</i>	<i>all 0s</i>	<i>Source</i>
<i>Host</i> específico nesta rede	<i>all 0s</i>	específico	<i>Destination</i>
<i>Loopback address</i>	127	qualquer	<i>Destination</i>

- » *Direct broadcast address* – identifica todos os *hosts* numa rede específica (*netid*), pelo que os pacotes com este endereço de destino devem ser encaminhados pelos *routers* no percurso
- » *Limited broadcast address* – endereço de *broadcast* na rede onde o pacote é gerado; os pacotes com este endereço de destino não são encaminhados para fora desta rede, sendo ignorados pelos *routers* respectivos

Interface de loopback

- Interface de teste, num computador
 - » O datagrama não é passado à carta de rede
 - » *netid* → 127
 - » Endereço IP da interface → por convenção, usa-se normalmente o endereço 127.0.0.1
 - » Nome → *localhost*
- Um datagrama enviado para *localhost* não é visto na rede
- Permite que um cliente e um servidor na mesma máquina comuniquem usando TCP/IP

IP – subredes

- Divisão de uma rede em subredes
 - » A parte correspondente ao endereço de um *host* passa a ter o significado de “identificador de subrede (*subnetid*) + identificador de *host* (*hostid*)”
 - » Exemplo: uma LAN / VLAN associada a uma subrede IP
 - » Todos os computadores de uma subrede têm o mesmo *subnetid* (e o mesmo *netid*)
- Permite construir múltiplas redes numa empresa
 - » Esconde para o exterior as redes da empresa
 - » Do exterior as redes da empresa são vistas como uma rede única (*netid*)
- Os *routers* da empresa fazem encaminhamento interno usando informação de subrede
- Máscara de subrede
 - » Identifica bits que devem ser interpretados como *netid* + *subnetid*

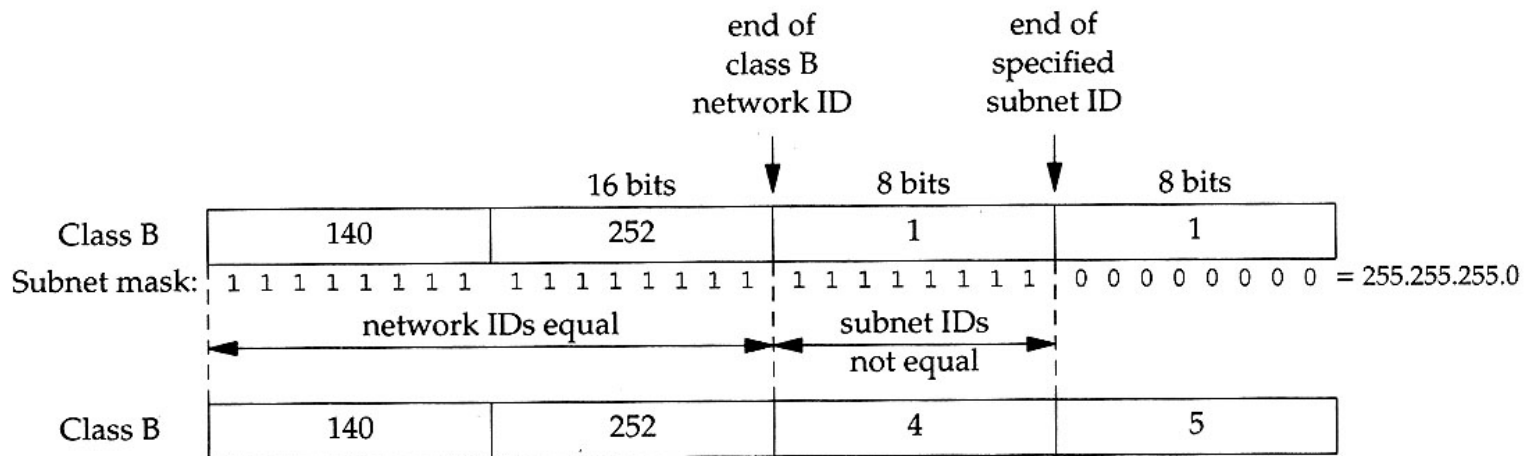
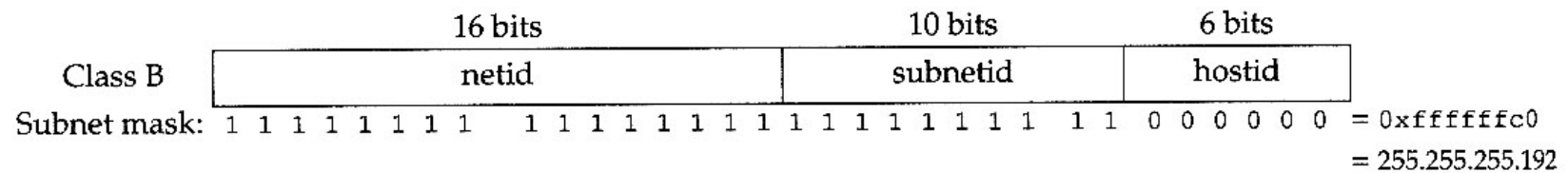
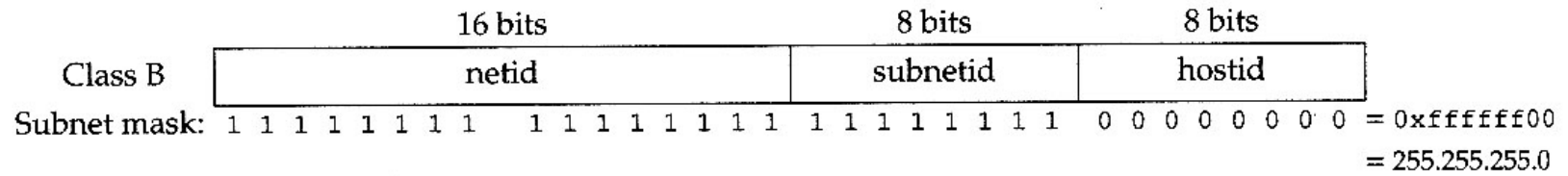
IP – máscaras

	Binary Representation	Dotted Decimal
IP address	11000000.11100100.00010001.00111001	192.228.17.57
Subnet mask	11111111.11111111.11111111.11100000	255.255.255.224
Bitwise AND of address and mask (resultant network/subnet number)	11000000.11100100.00010001.00100000	192.228.17.32
Subnet number	11000000.11100100.00010001.001	1
Host number	00000000.00000000.00000000.00011001	25

Default subnet masks

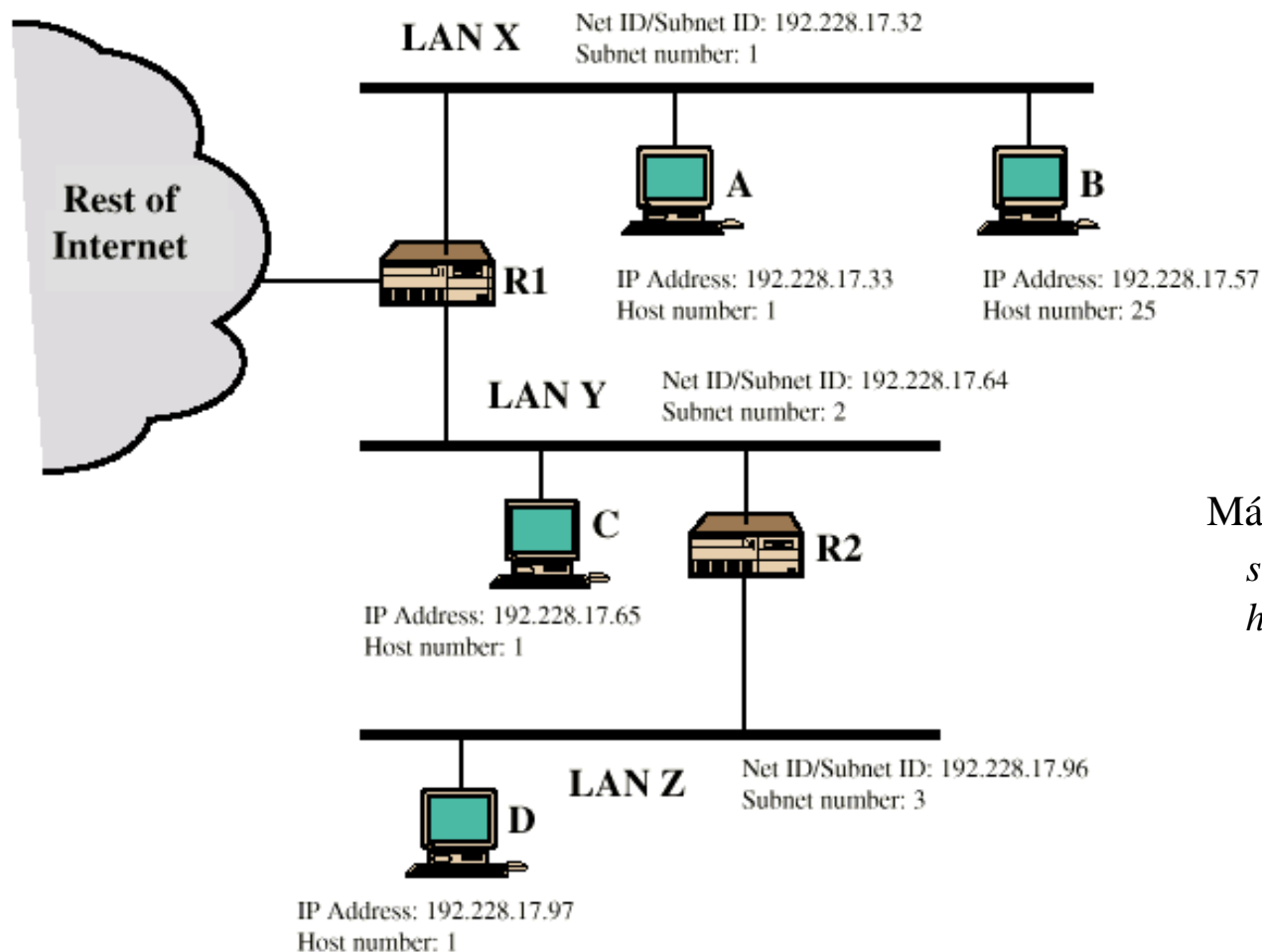
	Binary Representation	Dotted Decimal
Class A default mask	11111111.00000000.00000000.00000000	255.0.0.0
Example Class A mask	11111111.11000000.00000000.00000000	255.192.0.0
Class B default mask	11111111.11111111.00000000.00000000	255.255.0.0
Example Class B mask	11111111.11111111.11111000.00000000	255.255.248.0

IP – máscaras



Constituição de subredes

No exemplo supõe-se que a rede 192.228.17.0/24 é dividida em oito subredes constituídas com base numa máscara comum de 27 bits (mas poderiam formar-se subredes com máscaras diferentes)



Exemplos de subredes

192.228.17.0/27

192.228.17.32/27

192.228.17.64/27

192.228.17.96/27

.....

192.228.17.224/27

Máscara de subrede – 27 bits

subnetid – 3 bits (8 subredes)

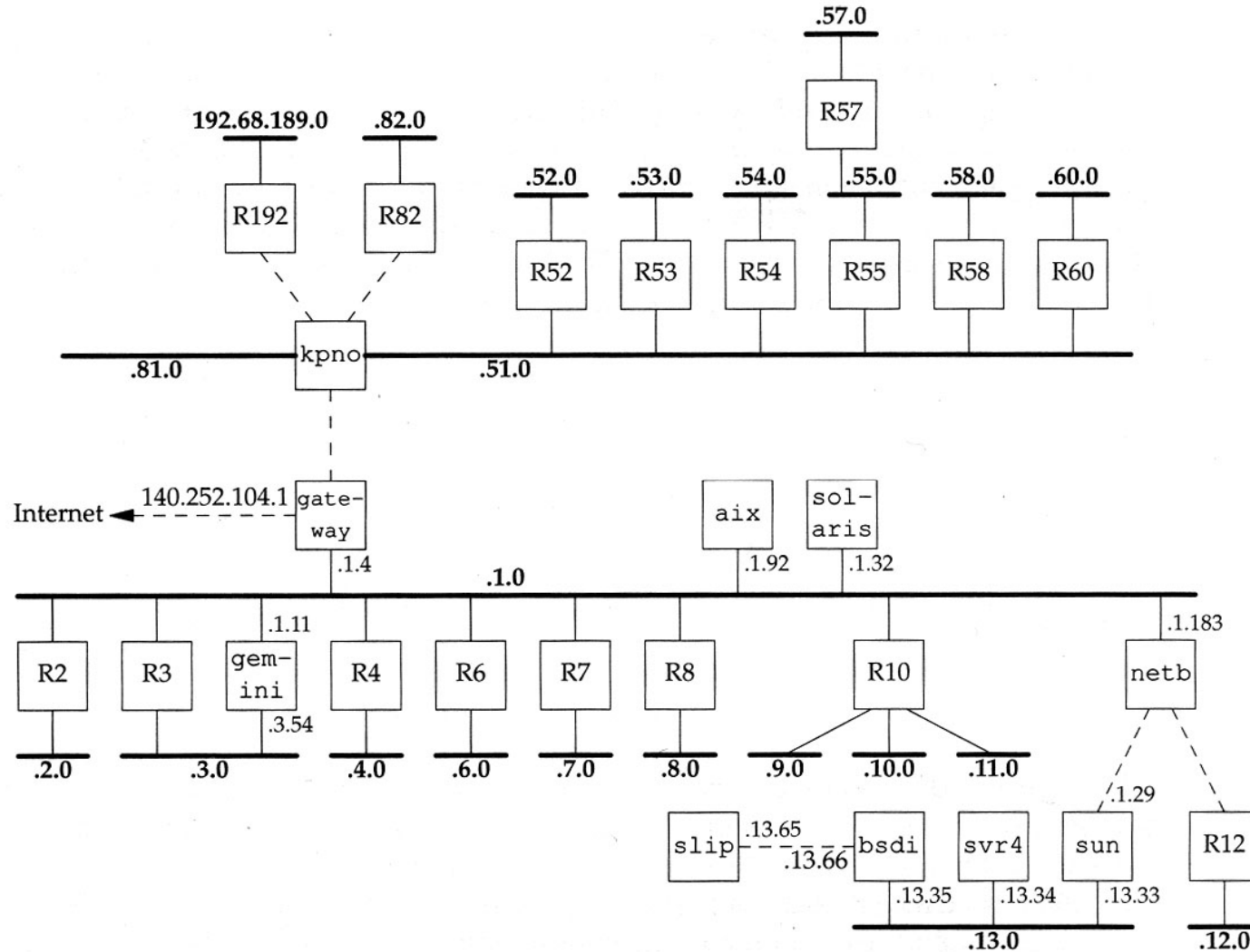
hostid – 5 bits

até 30 *hosts* por subrede

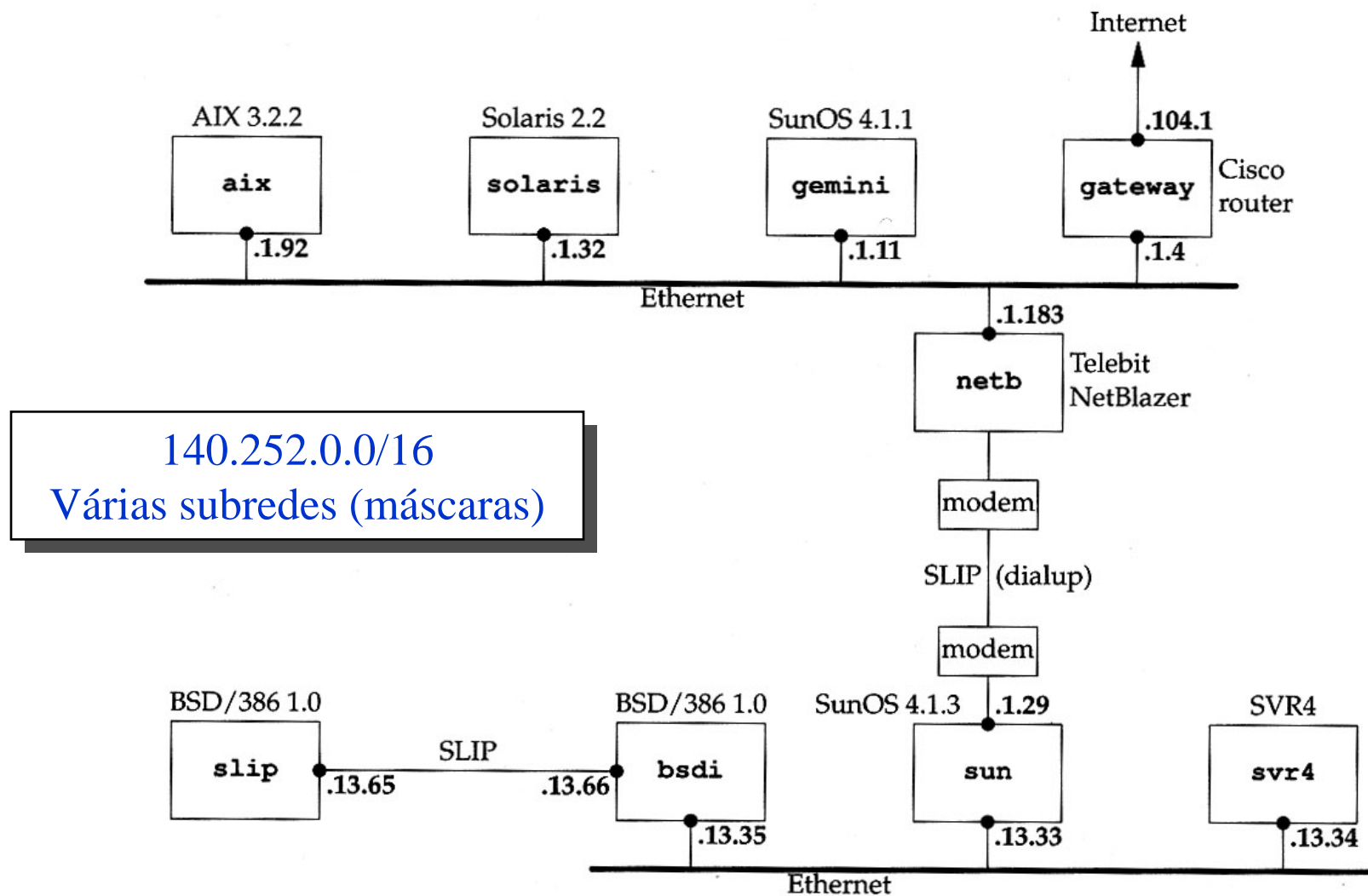
all 0 – identifica subrede

all 1 – endereço de *broadcast*

Exemplo de subredes



Rede exemplo



Encaminhamento IP

- Cada sistema IP (*host* ou *router*) deve possuir uma tabela de encaminhamento; a tabela é relativamente simples no caso de um *host*, que pode ser configurado para funcionar, em certos casos, como *router*
 - » No caso de o sistema de destino estar directamente ligado ao *host* de origem (e.g., uma ligação ponto a ponto) ou numa rede partilhada (e.g., uma LAN), os pacotes IP são enviados directamente para o destino
 - » Caso contrário, os pacotes são enviados pelo *host* de origem para um *router* (*next hop router*) na mesma rede – a existência de mais do que um *router* impõe que seja seleccionado um deles, de acordo com o destino dos pacotes
- Cada entrada de uma tabela de encaminhamento contém
 - » Um endereço IP, que pode ser um *host address* (*netid* + *hostid*) ou um *network address* (*netid*, *hostid* = 0) – estes casos são distinguidos por uma *flag* H
 - » A indicação de uma rota, que pode ser indirecta (endereço IP de um *next hop router*) ou directa (endereço IP de uma interface local) – estes casos são distinguidos por uma *flag* G
 - » Interface de rede à qual o datagrama deve ser passado

Encaminhamento IP

- Para encaminhar um pacote, a tabela é pesquisada com o objectivo de encontrar
 - » Uma entrada que seja igual ao endereço de destino do pacote (*netid + hostid*)
 - Ligações ponto a ponto são um exemplo deste caso
 - » Uma entrada que seja igual ao *netid* de destino, no caso de o passo anterior não ter sucesso
 - Todas as máquinas numa LAN são cobertas por este caso
 - » Uma entrada por omissão (*default*), no caso de não existir qualquer entrada que satisfaça os critérios anteriores
- Se for encontrada uma entrada que satisfaça uma das condições, o pacote é enviado de acordo com o especificado nessa entrada (*flag G*)
 - » Para um *next hop router* – o endereço IP do pacote é o da máquina de destino, mas o endereço de nível 2 (trama) é o do *next hop router* (e.g., um endereço MAC numa LAN)
 - » Para um destino directamente acessível – o endereço IP do pacote e o endereço de nível 2 (trama) são endereços da máquina de destino
- Se não for satisfeita qualquer das condições (o que significa que não existe uma entrada *default*), o pacote não é encaminhável, sendo por isso descartado

Endereçamento hierárquico – agregação de rotas

- O endereçamento hierárquico permite divulgação eficiente de informação de encaminhamento (agregação de rotas), conforme ilustrado
 - » O ISP 1 dispõe de endereços da rede 200.23.16.0/20 (200.23.16.0 a 200.23.31.255) e dividiu esse espaço para atribuição de endereços a 8 organizações (conforme ilustrado)
 - » O ISP2 dispõe de endereços da rede 199.31.0.0/16

Organização 0

200.23.16.0/23

Organização 1

200.23.18.0/23

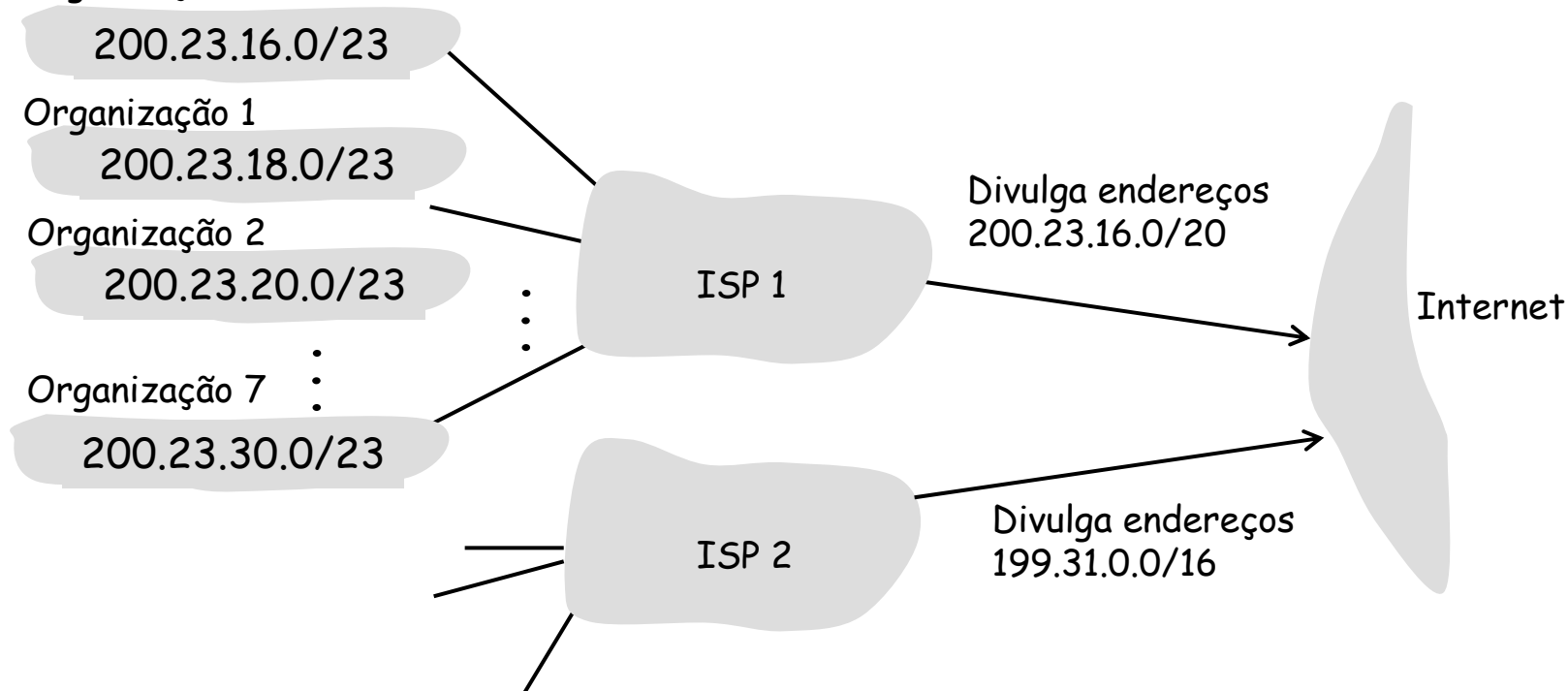
Organização 2

200.23.20.0/23

⋮

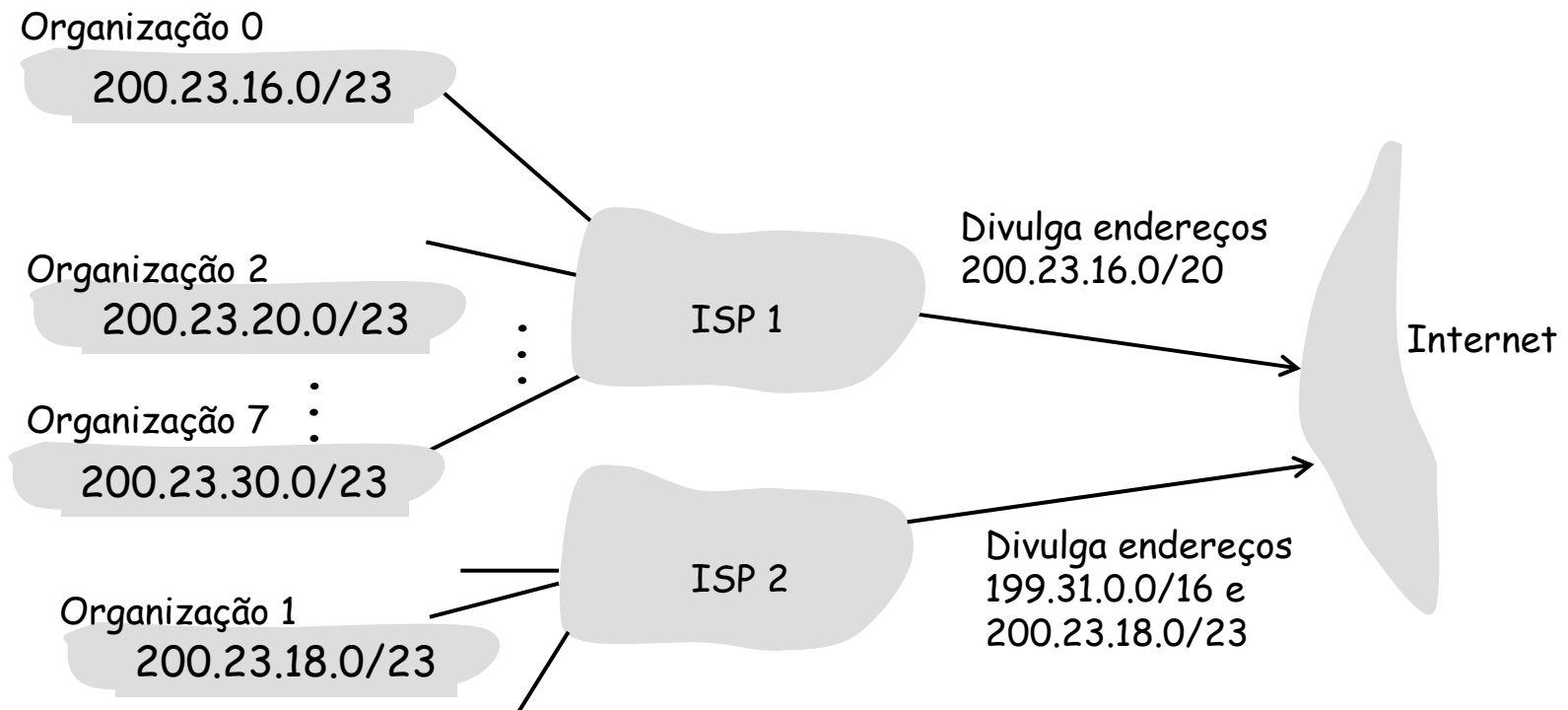
Organização 7

200.23.30.0/23



Endereçamento hierárquico – rotas específicas

- Pode ser necessário divulgar rotas específicas, no caso de a agregação de rotas não ser (ou deixar de ser) possível (no exemplo, a Organização 1 alterou o seu ISP, mas manteve os endereços originais)
- Esta situação determina que a selecção de uma entrada na tabela de encaminhamento (comparação do endereço de destino com os endereços presentes na tabela) deve ser feita com base na coincidência com o prefixo mais longo (*longest prefix matching*)



Exemplos de tabelas de encaminhamento

Host 140.252.13.34 (svr4) na rede 140.252.13.32/27

Destination	Gateway	Flags	Refcnt	Use	Interface
140.252.13.65	140.252.13.35	UGH	0	0	emd0
127.0.0.1	127.0.0.1	UH	1	0	lo0
default	140.252.13.33	UG	0	0	emd0
140.252.13.32	140.252.13.34	U	4	25043	emd0

Host / router 140.252.13.33 (sun) na rede 140.252.13.32/27

Destination	Gateway	Flags	Refcnt	Use	Interface
140.252.13.65	140.252.13.35	UGH	0	171	le0
127.0.0.1	127.0.0.1	UH	1	766	lo0
140.252.1.183	140.252.1.29	UH	0	0	s10
default	140.252.13.183	UG	1	2955	s10
140.252.13.32	140.252.13.33	U	8	99551	le0

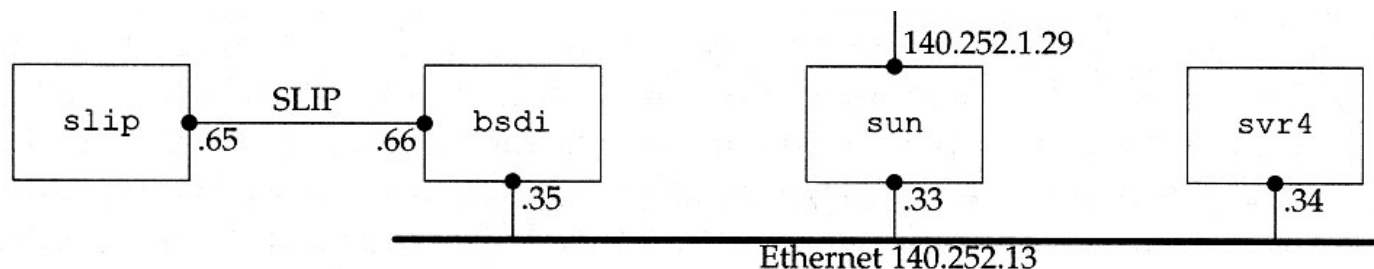


Tabela de encaminhamento – interpretação

Destination	Gateway	Flags	Refcnt	Use	Interface
140.252.13.65	140.252.13.35	UGH	0	0	emd0
127.0.0.1	127.0.0.1	UH	1	0	lo0
default	140.252.13.33	UG	0	0	emd0
140.252.13.32	140.252.13.34	U	4	25043	emd0

Host 140.252.13.34 (svr4) na rede 140.252.13.32/27 (*subnetid* – 3 bits; *hostid* – 5 bits)

Gateways 140.252.13.33 e 140.252.13.35

Flags

U *route Up*

G *route to a Gateway (next hop router)*

se omitido: rota directa; o endereço da *Gateway* é o endereço IP da interface local do *host* svr4

H *route to a Host* – o endereço IP de destino na tabela é um *host address* completo

se omitido: *route to a network* – o endereço IP de destino na tabela é um *network address*

Refcnt

Número de utilizações activas da rota (por exemplo ligações TCP estabelecidas)

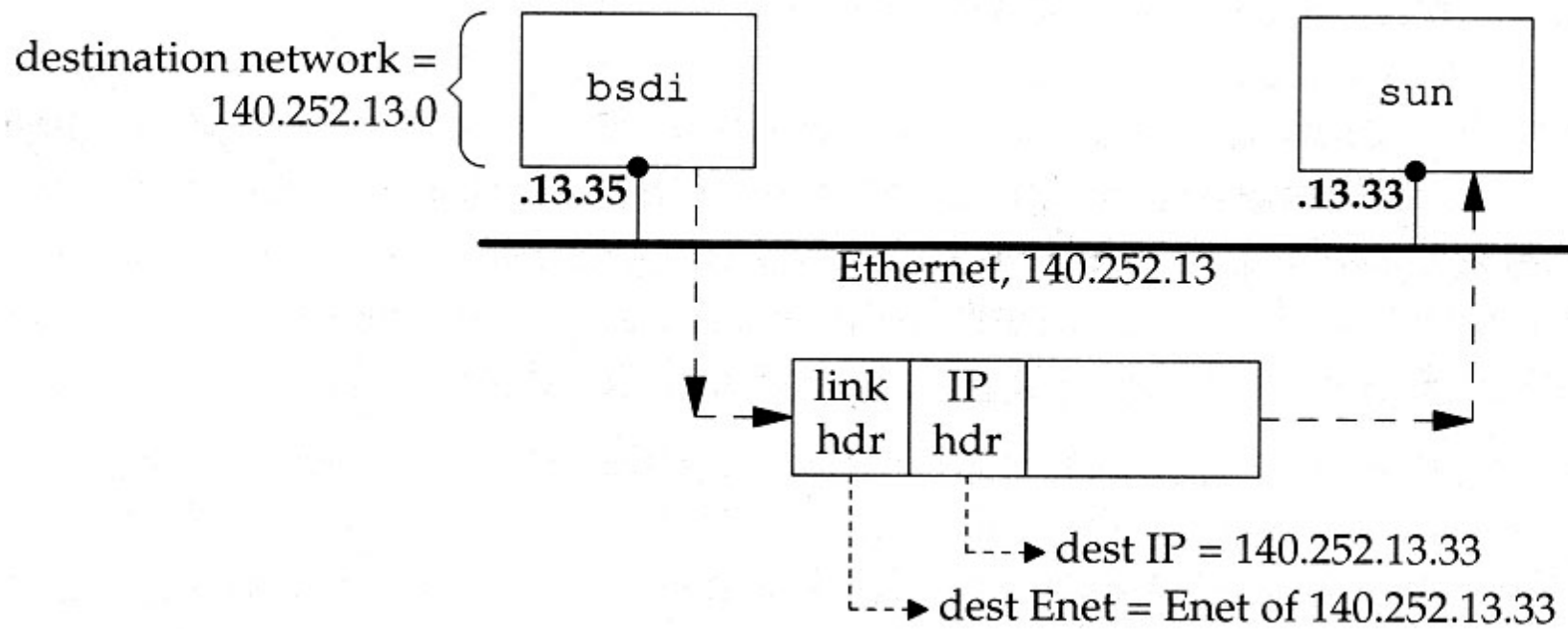
Use

Número de pacotes enviados pela rota

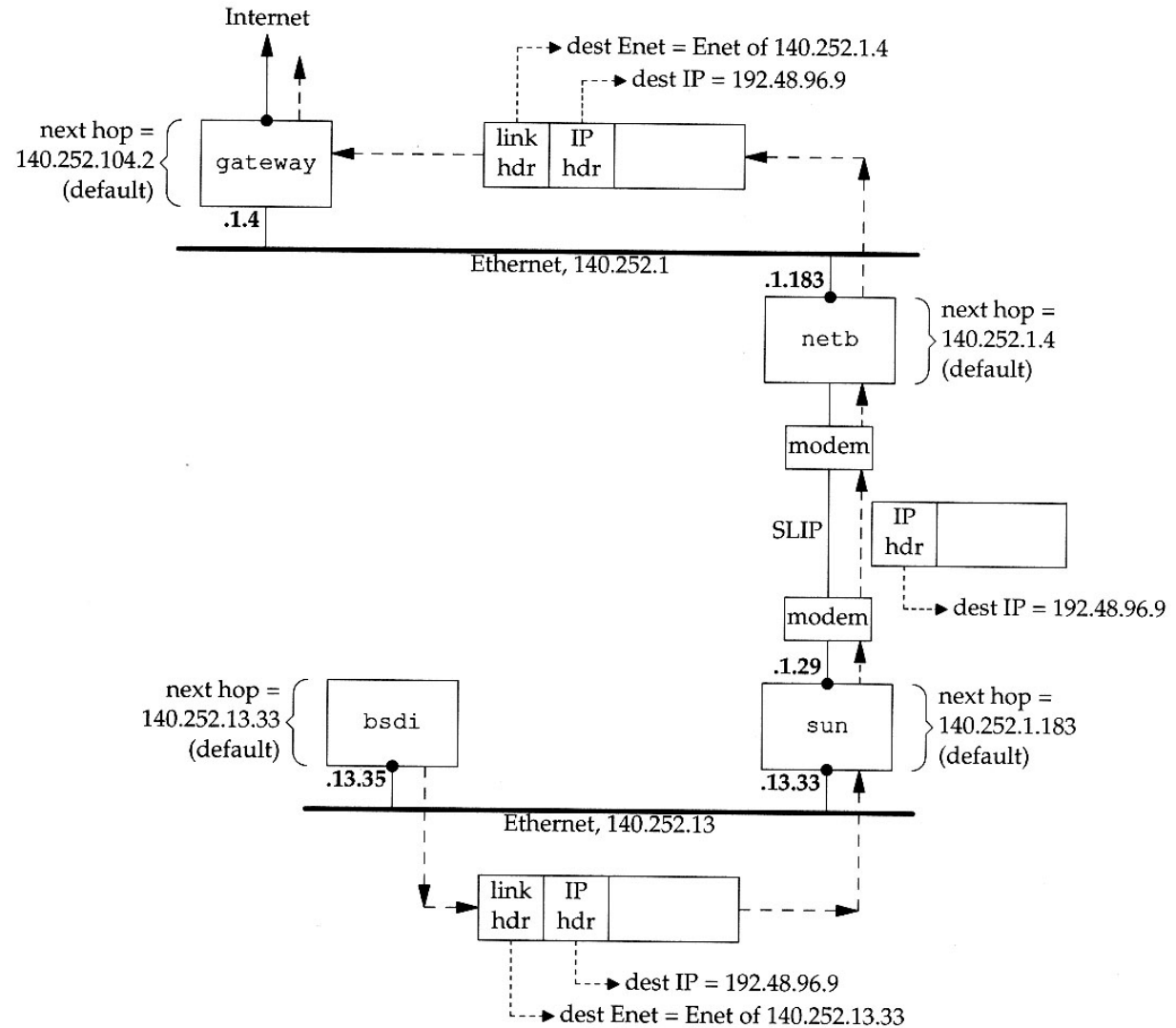
Interface

Nome da interface local

Exemplo 1

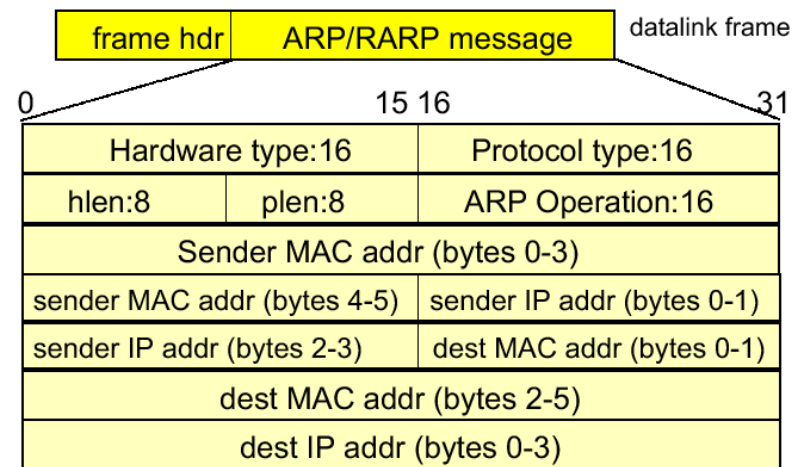


Exemplo 2



ARP – Address Resolution Protocol

- Host TCP/IP → endereço IP
- Carta de rede → endereço físico (*hardware*)
 - » Em LANs – endereço MAC
- ARP
 - » Mapeamento dinâmico
 - Endereço IP ↔ endereço de *hardware*



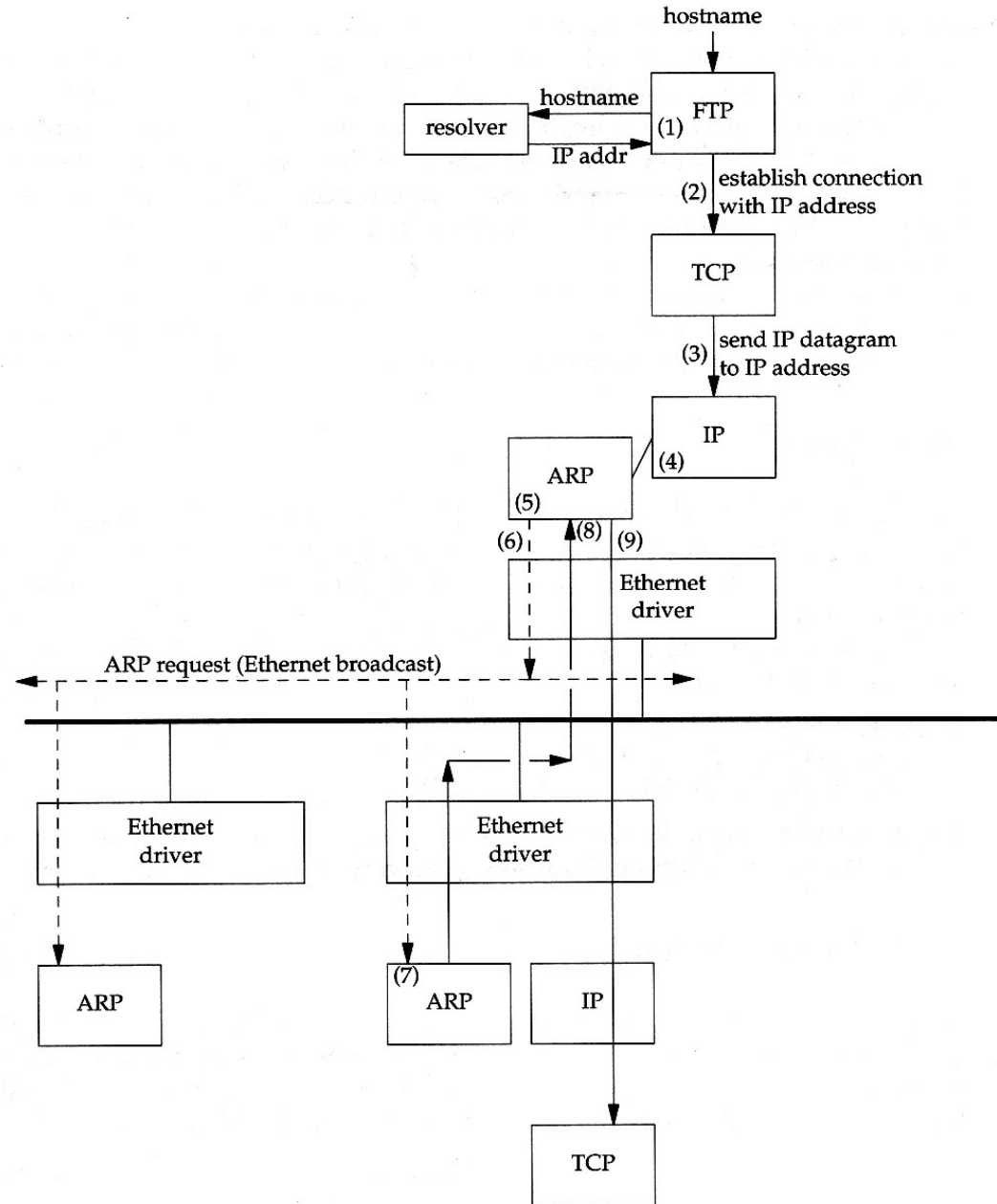
32-bit Internet address



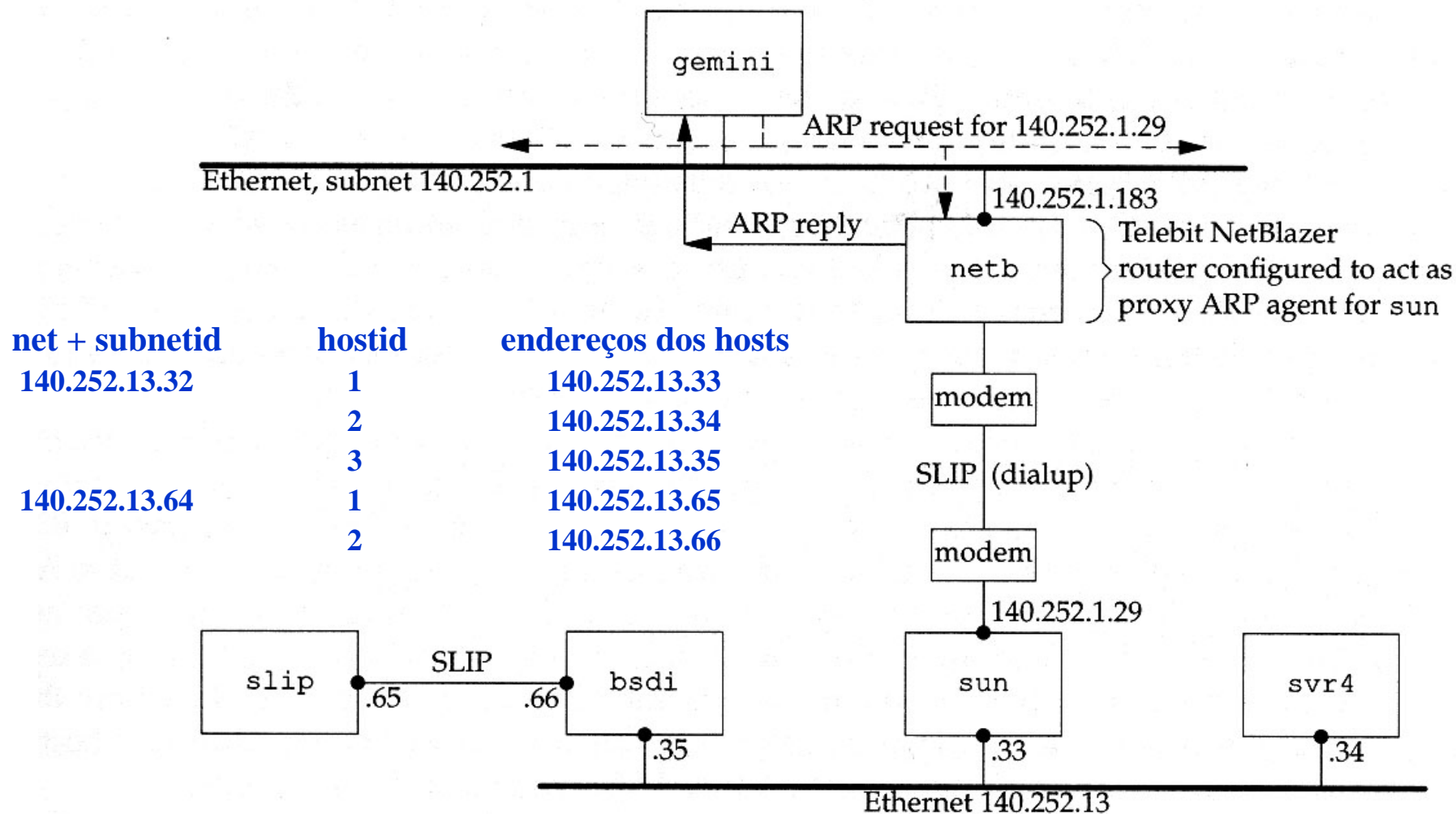
48-bit Ethernet address

- **hardware type** : Ethernet=1 ARCNET=7, localtalk=11
- **protocol type** : IP=0x800
- **hlen** : length of hardware address, Ethernet=6 bytes
- **plen** : length of protocol address, IP=4 bytes
- **ARP operation** : ARP request = 1, ARP reply = 2
RARP request = 3, RARP reply = 4

Exemplo

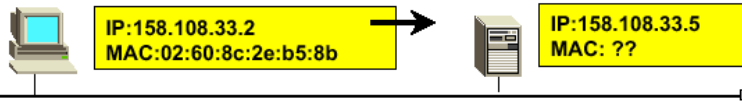


Proxy ARP



ARP Request e ARP Reply

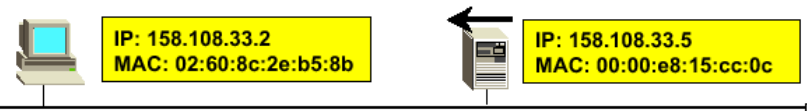
ARP request packet



Sample ARP request Ethernet packet

FF:FF:FF:FF:FF:FF			dest MAC (broadcast)
02:60:8c:2e:b5:8b			source MAC
0x0806			ARP frame type
0x01			Ethernet / IP
0x06	0x04	0x001	MAC=6/ IP=4 /request
02:60:8c:2e:b5:8b			source MAC
158.108.33.2			source IP
00:00:00:00:00:00			dest MAC (unknown)
158.108.33.5			dest IP
checksum			Ethernet checksum

ARP reply packet



Sample ARP reply Ethernet packet

02:60:8c:2e:b5:8b			dest MAC (unicast)
00:00:e8:15:cc:0c			source MAC
0x0806			ARP frame type
0x01			Ethernet / IP
0x06	0x04	0x002	MAC=6/ IP=4 /reply
00:00:e8:15:cc:0c			source MAC
158.108.33.5			source IP
02:60:8c:2e:b5:8b			dest MAC
158.108.33.2			dest IP
checksum			Ethernet checksum

ifconfig

```
sun% /usr/etc/ifconfig -a
```

```
le0 : flags=63<UP, BROADCAST, NOTRAILERS, RUNNING>  
inet 140.252.13.33 netmask fffffffe0 broadcast 140.252.13.63
```

```
slo : flags=105KUP, POINTOPOINT, RUNNING, LINKO>  
inet 140.252.1.29 -> 140.252.1.183 netmask fffffff0
```

```
lo0: flags=49<UP, LOOPBACK, RUNNING>  
inet 127.0.0.1 netmask ff000000
```

```
bsdi% /sbin/ifconfig we0
```

```
we0: flags=863<UP, BROADCAST, NOTRAILERS, RUNNING, SIMPLEX>  
inet 140.252.13.35 netmask fffffffe0 broadcast 140.252.13.63
```

```
bsdi% /sbin/ifconfig slo
```

```
slo : flags=1011<UP, POINTOPOINT, LINKO  
inet 140.252.13.66 -> 140.252.13.65 netmask fffffffe0
```

```
slip% /sbin/ifconfig slo
```

```
slo : flags=1011<UP, POINTOPOINT, LINKO  
inet 140.252.13.65 --> 140.252.13.66 netmask fffffffe0
```

```
svr4% /usr/sbin/ifconfig emd0
```

```
emd0: flags=23<UP, BROADCAST, NOTRAILERS>  
inet 140.252.13.34 netmask fffffffe0 broadcast 140.252.13.63
```

netstat

```
sun% netstat -in
```

Name	Mtu	Net/Dest	Address	lpkts	lerrs	Opkts	Oerrs	Collis	Queue
le0	1500	140.252.13.32	140.252.13.33	67719	0	92133	0	1	0
slo	552	140.252.1.183	140.252.1.29	48035	0	54963	0	0	0
lo0	1536	127.0.0.0	127.0.0.1	15548	0	15548	0	0	0

ICMP – Internet Control Message Protocol

- O ICMP está descrito no RFC 792
- O ICMP transfere mensagens de controlo
 - » Entre um *host* e um *router* ou entre dois *hosts*
 - » As mensagens ICMP indicam problemas relacionados com o transporte de datagramas
- Mensagens ICMP são encapsuladas em datagramas IP (não fiável)

ICMP – tipos de mensagem

0	8	16	31
Type	Code	Checksum	
Unused			
IP Header + 64 bits of original datagram			

(a) Destination Unreachable; Time Exceeded; Source Quench

0	8	16	31
Type	Code	Checksum	
Identifier		Sequence Number	
Originate Timestamp			

(e) Timestamp

0	8	16	31
Type	Code	Checksum	
Pointer	Unused		
IP Header + 64 bits of original datagram			

(b) Parameter Problem

0	8	16	31
Type	Code	Checksum	
Identifier		Sequence Number	
Originate Timestamp			
Receive Timestamp			
Transmit Timestamp			

(f) Timestamp Reply

0	8	16	31
Type	Code	Checksum	
Gateway Internet Address			
IP Header + 64 bits of original datagram			

(c) Redirect

0	8	16	31
Type	Code	Checksum	
Identifier		Sequence Number	

(g) Address Mask Request

0	8	16	31
Type	Code	Checksum	
Identifier		Sequence Number	
Optional data			

(d) Echo, Echo Reply

0	8	16	31
Type	Code	Checksum	
Identifier		Sequence Number	
Address Mask			

(h) Address Mask Reply

Ping

- ICMP *Echo / Echo Reply*
- *Echo*
 - » Identificador
 - Pid do cliente
 - » Sequência
 - Começa em 0
 - » Dados
 - Tempo de envio
- *Echo Reply*
 - » Resposta enviada por servidor *ping (kernel)*
 - » Ecoa tudo

```
sun% ping gemini
PING gemini: 56 data bytes
64 bytes from gemini (140.252.1.11): icmp_seq=0. time=373. ms
64 bytes from gemini (140.252.1.11): icmp_seq=1. time=360. ms
64 bytes from gemini (140.252.1.11): icmp_seq=2. time=340. ms
64 bytes from gemini (140.252.1.11): icmp_seq=3. time=320. ms
64 bytes from gemini (140.252.1.11): icmp_seq=4. time=330. ms
64 bytes from gemini (140.252.1.11): icmp_seq=5. time=310. ms
64 bytes from gemini (140.252.1.11): icmp_seq=6. time=290. ms
64 bytes from gemini (140.252.1.11): icmp_seq=7. time=300. ms
64 bytes from gemini (140.252.1.11): icmp_seq=8. time=280. ms
64 bytes from gemini (140.252.1.11): icmp_seq=9. time=290. ms
64 bytes from gemini (140.252.1.11): icmp_seq=10. time=300. ms
64 bytes from gemini (140.252.1.11): icmp_seq=11. time=280. ms
--gemini PING Statistics--
12 packets transmitted, 12 packets received, 0% packet loss
round-trip (ms) min/avg/max = 280/314/373
```

Traceroute

- Permite detectar rotas entre máquinas
- Usa datagramas IP e mensagens de erro ICMP
 - » *Traceroute* começa por enviar um datagrama para o destino pretendido, com TTL = 1 e porta UDP inexistente
 - O primeiro *router*
 - Decrementa valor de TTL → elimina datagrama (TTL = 0)
 - Envia mensagem de erro ICMP (*time exceeded*) para origem
 - *Traceroute* obtém a sua identificação
 - » *Traceroute* envia novo datagrama para o mesmo destino, com TTL = 2 e porta UDP inexistente
 - O segundo *router* repete o processo e assim sucessivamente
 - » *Traceroute* continua a enviar datagramas com TTL crescente até se atingir a máquina de destino
 - A máquina de destino, após receber o datagrama, envia
 - Mensagem de erro ICMP – *port unreachable*
 - *Traceroute* obtém a sua identificação

Traceroute – exemplo

```
svr4% traceroute slip
```

```
traceroute to slip (140.252.13.65), 30 hops max, 40 byte packets  
1 bsdi (140.252.13.35) 20 ms 10 ms 10 ms  
2 slip (140.252.13.65) 120 ms 120 ms 120 ms
```

```
slip% traceroute svr4
```

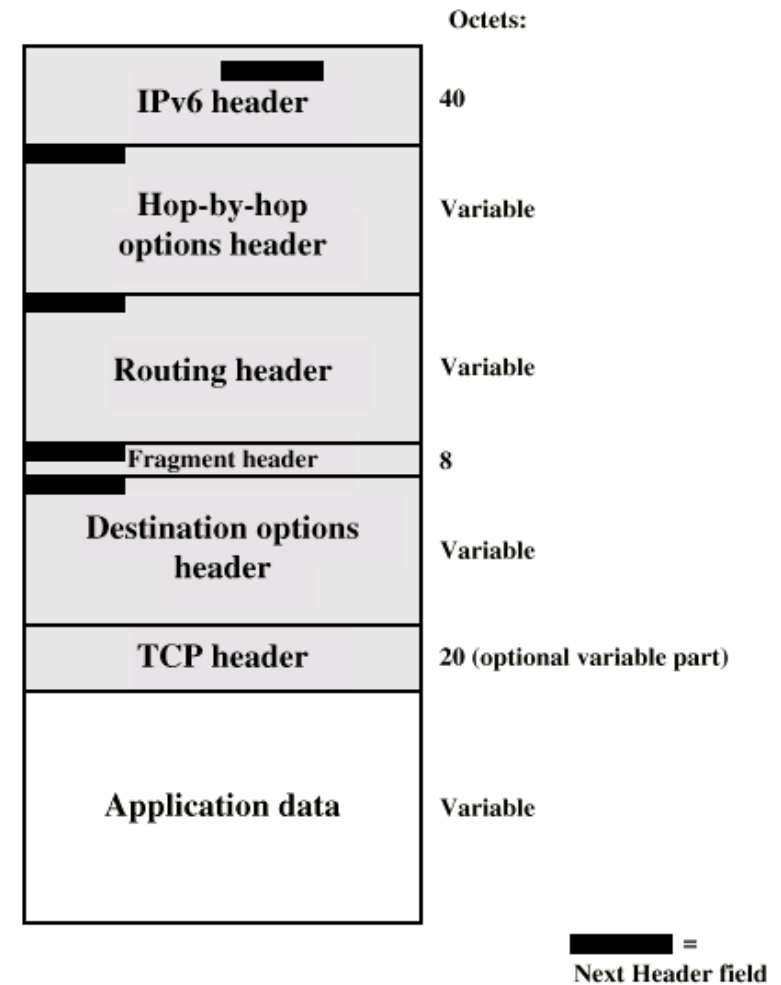
```
traceroute to svr4 (140.252.13.34), 30 hops max, 40 byte packets  
1 bsdi (140.252.13.66) 110 ms 110 ms 110 ms  
2 svr4 (140.252.13.34) 110 ms 120 ms 110 ms
```

Versões IP – (IPv4 e IPv6)

- Versões
 - » IP v1-v3 → fora de uso
 - » IP v4 → versão em uso
 - » IP v5 → orientado às ligações (nunca foi implementado)
 - » IP v6 → substituto do IP v4
- Limitações do IPv4
 - » Endereçamento
 - Dois níveis de endereçamento (rede, *host*) desperdiça endereços
 - Endereços usados mesmo quando a rede não está ligada à Internet
 - Crescimento do número de redes e da Internet
 - Um único endereço por interface
 - » Não suporta requisitos de novos tipos de serviços
- Melhorias do IPv6
 - » Flexibilidade de endereçamento
 - Maior espaço de endereçamento (128 bits)
 - *Multicast* – mais versátil e escalável
 - » Reserva de recursos
 - Substituição do *Type of Service*
 - Introdução do conceito de fluxo de pacotes
 - Fluxos com tratamento especial (QoS), e.g., vídeo em tempo real

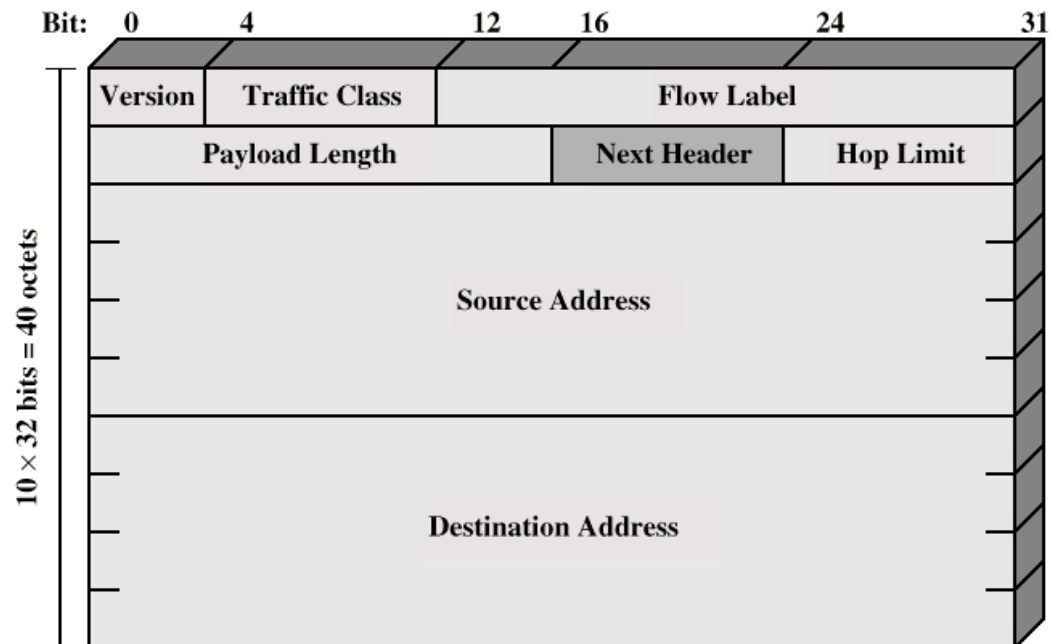
IPv6 – protocolo

- Cabeçalho IPv6
- Cabeçalhos secundários (de extensão)
 - » *Hop-by-Hop Options*
 - Processado e alterado nos *routers*
 - » *Routing*
 - Semelhante ao *Source Routing* do IPv4
 - » *Fragment*
 - » *Destination options*
 - Para nó de destino



Cabeçalho IPv6

- *Version*
 - » 6 (IPv6)
- *Traffic Class*
 - » Classes / prioridades de pacote
- *Flow Label*
 - » Suporte de QoS
- *Payload length*
 - » Cabeçalhos secundários + dados
- *Next Header*
 - » Tipo do próximo cabeçalho
(Extensão ou nível superior)
- *Source Address*
- *Destination Address*



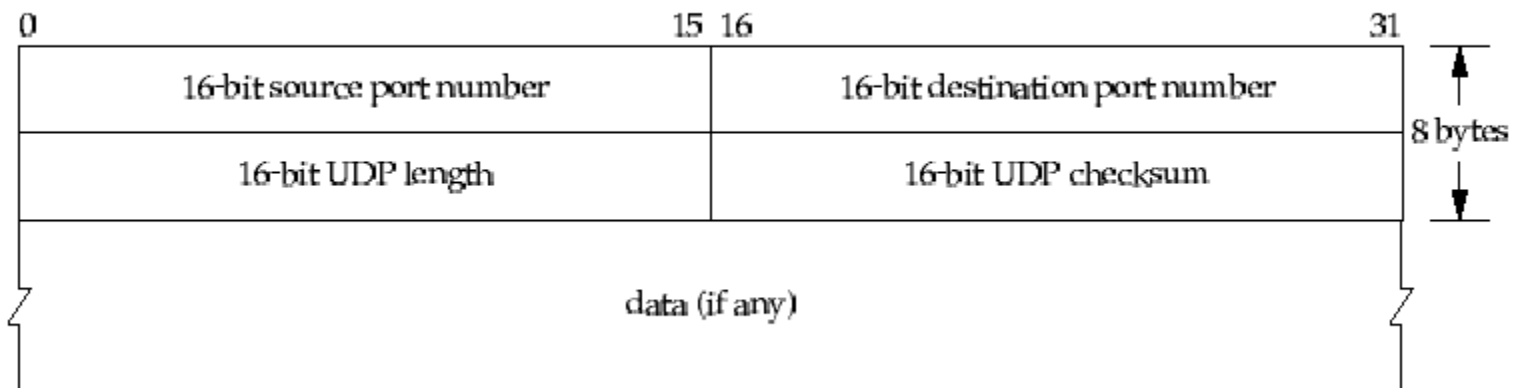
IPv6 – endereços

- Endereços com 128 bits
- Atribuídos a interfaces
- Uma interface pode ter múltiplos endereços
- Tipos de endereços
 - » *Unicast*
 - Endereço específico (individual) atribuído a uma interface
 - » *Anycast*
 - Endereço associado a um conjunto de interfaces
 - Pacote entregue a qualquer interface (a mais próxima)
 - » *Multicast*
 - Endereço associado a um conjunto de interfaces
 - Pacote entregue a todas as interfaces

UDP – User Datagram Protocol

- RFC 768
- Características
 - » Protocolo de transporte, não orientado às conexões (*connectionless*)
 - » Serviço de entrega de pacotes não fiável
 - » Usa serviços IP
 - » Multiplexagem de vários fluxos UDP sobre o mesmo endereço IP

UDP Header

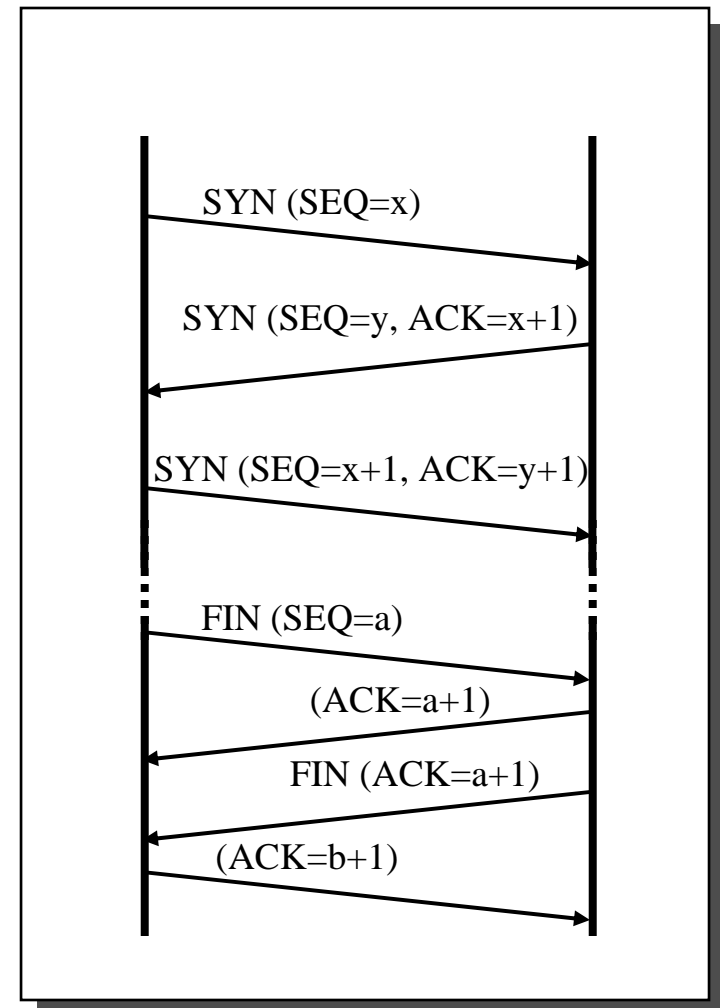


UDP Length – comprimento total do pacote

UDP Checksum – opcional

TCP – Transmission Control Protocol

- Descrito no RFC 793
- Principais características
 - » Assegura um fluxo de octetos extremo a extremo, fiável, sobre um suporte não fiável
 - » Protocolo orientado às conexões
 - Estabelecimento de conexão TCP – *three way handshake*
 - Conexões *full-duplex*
 - » Controlo de erros
 - Confirmação positiva (ACK)
 - Recupera de perdas e erros (retransmissões) após *time-out*
 - Entrega ordenada dos dados à aplicação
 - » Controlo de fluxo e de congestionamento
 - Mecanismo de janela
 - » Multiplexagem de várias conexões TCP sobre o mesmo endereço IP



TCP – Transmission Control Protocol

Source Port – porta de origem

Destination Port – porta de destino

Sequence Number – identifica, no fluxo do emissor, a sequência de octetos enviada

Acknowledgement Number – corresponde ao número do octeto que se espera receber

HLEN – comprimento do cabeçalho TCP (em palavras de 32 bits)

URG – informa se o campo *Urgent Pointer* deve ser interpretado

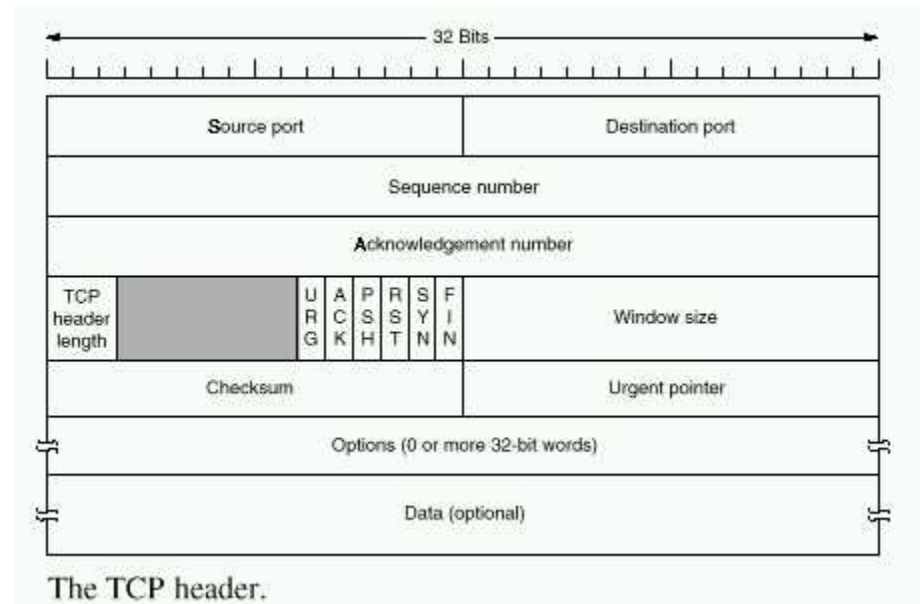
ACK – informa se o campo *Ack Nbr* é válido

PSH – permite forçar o envio imediato de dados (sem esperar dados adicionais)

RST – usado para reinicializar uma conexão

SYN – permite estabelecer uma conexão

FIN – permite terminar uma conexão



Window Size – número de octetos que o par (*peer*) da comunicação pode enviar sem confirmação (controlo de fluxo)

Checksum – abrange o cabeçalho, os dados e o pseudo-cabeçalho

Controlo de fluxo e de congestionamento

- Quando dois sistemas (*hosts*) comunicam através duma rede, o receptor deve poder regular o débito do emissor – este controlo é realizado extremo-a-extremo, tipicamente na camada de Transporte, sendo o TCP um exemplo de referência
- Por outro lado, o débito de um fluxo TCP que a rede é capaz de sustentar depende da capacidade das ligações ao longo do percurso (e respectiva carga) e dos recursos de transmissão que é possível atribuir ao fluxo (em competição com outros fluxos)
- O controlo de fluxo pelo receptor torna-se efectivo quando a rede pode suportar o débito da fonte, devendo este convergir para o valor do débito permitido pelo receptor
 - » O controlo de fluxo extremo-a-extremo é influenciado pelo comportamento dinâmico da rede – o atraso de ida e volta (*round trip time*) e o débito dos fluxos dependem de variações quer do tráfego submetido à rede quer do estado da rede
- Se o estrangulamento se dever à rede e não ao receptor (devido à capacidade limitada de algumas ligações ou da sobrecarga de nós), o débito é na prática limitado pela rede
 - » Caso os emissores não sejam capazes de adaptar o débito às condições da rede, ocorrerá perda de pacotes quando se exceda a capacidade dos *buffers* nalgum nó
 - » Em caso de congestionamento são necessárias medidas adicionais e os emissores devem reduzir o débito com base em informação implícita ou explícita do estado da rede
- Os mecanismos de controlo de fluxo devem regular os débitos de fontes de tráfego de modo que convirjam para os débitos de serviço suportados pelos receptores e pela rede

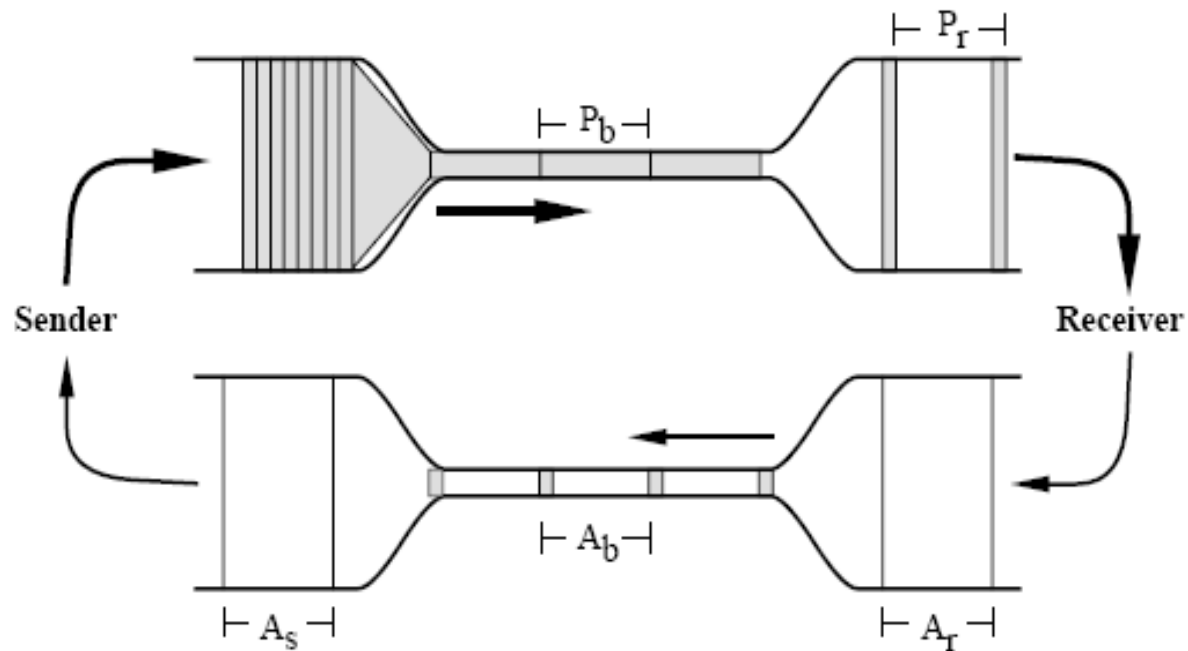
TCP – janelas de controlo

- O TCP realiza controlo de congestionamento extremo-a-extremo, por meio duma janela dinâmica que é actualizada com base na janela do receptor e em informação implícita do estado da rede
- Em primeiro lugar o emissor deve respeitar a janela anunciada pelo receptor
 - » Para um dado número de pacotes em trânsito, cada ACK recebido indica que um pacote deixou a rede e um novo pacote pode ser enviado (rotação da janela)
 - » Uma vez que o ritmo a que os ACKs são recebidos (e, por isso, novos pacotes podem ser enviados) é determinado pelo estado da rede, o TCP é *self-clocking*
- Contudo, a dinâmica da rede é complexa – o número de sessões activas e a capacidade disponível variam no tempo e a rede pode ficar congestionada
- O TCP deixa que cada fonte determine de forma independente a capacidade que pode usar e adapte o seu débito ao estado da rede, por meio duma janela de congestionamento dinâmica (*congestion window*) actualizada com base na percepção do nível de congestionamento actual da rede
- A janela de congestionamento (*cwnd*) representa controlo de fluxo realizado pelo emissor, enquanto o receptor impõe controlo de fluxo por meio da janela que anuncia (*rwnd*) – a janela de transmissão é o menor dos dois valores

$$\text{current sender window} = \min(\text{cwnd}, \text{rwnd})$$

TCP – auto sincronização (*self-clocking*)

- P_b representa o menor espaçamento entre pacotes de dados na ligação mais lenta (*bottleneck*), pelo que, no receptor, $P_r = P_b$
- Assumindo o mesmo tempo de processamento para todos os pacotes, o espaçamento entre ACKs gerados pelo receptor é $A_r = P_r = P_b$ e, considerando condições idênticas no percurso inverso (e que ACKs ocupam menos largura de banda que pacotes de dados), então $A_s = A_b = A_r = P_b$
- Se, após um *burst* inicial, só forem enviados novos pacotes de dados após recepção de ACKs, o espaçamento de pacotes no emissor coincide com o tempo entre pacotes na ligação mais lenta



TCP – algoritmos de controlo

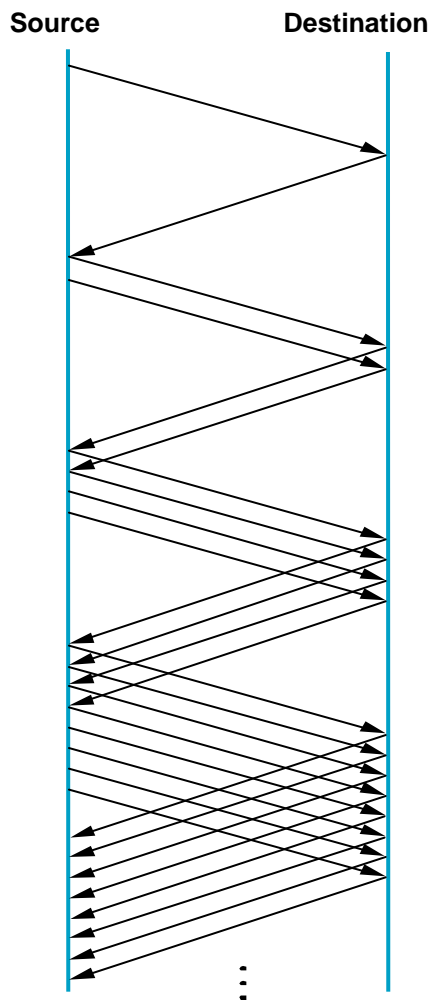
- O TCP implementa quatro algoritmos, habitualmente combinados em dois grupos – *slow start* e *congestion avoidance* por um lado, e *fast retransmit* e *fast recovery* por outro
- O emissor TCP mantém a informação seguinte (por conexão)
 - O estado de duas janelas (*cwnd* e *rwnd*) – o mínimo destes valores é a sua janela actual (*sender window*)
 - O valor de um limiar designado *slow start threshold* (*ssthresh*), que permite decidir se deve ser usado *slow start* (janela abaixo do limiar) ou *congestion avoidance* (janela acima do limiar)
 - Uma variável *FlightSize*, que indica a quantidade de dados enviados mas não confirmados (o seu valor é limitado pelo tamanho actual da janela)
- Na descrição dos algoritmos e actualização das variáveis considera-se também o tamanho máximo dos segmentos que o emissor pode transmitir – *Sender Maximum Segment Size* (SMSS)

TCP – slow start e congestion avoidance

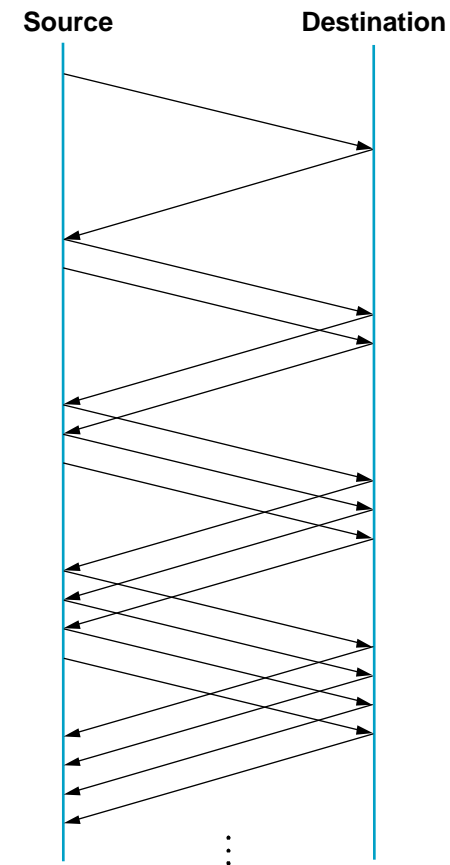
- O TCP assume que a perda de um segmento se deve a congestionamento da rede e o emissor usa dois indicadores para o efeito – *time-out* e recepção de ACKs duplicados, que desencadeiam acções de recuperação diferentes
- Após o estabelecimento de uma conexão ou ocorrência de *time-out*, o valor de *cwnd* é fixado em um segmento (SMSS) e o emissor entra no modo *slow start*
- No modo *slow start*, *cwnd* é incrementado no máximo de SMSS bytes por cada ACK recebido – isto significa que o valor de *cwnd* duplica por cada *round trip time* (*exponential increase*)
- Quando *cwnd* atingir o valor actual de *ssthresh*, o emissor comuta para o modo *congestion avoidance* e *cwnd* é incrementado de um segmento (SMSS) por cada *round trip time* (*linear increase*)
 - » Para o efeito, *cwnd* pode ser actualizado por cada novo ACK recebido (não duplicado) usando a fórmula
$$cwnd = cwnd + SMSS * SMSS / cwnd$$
- *Congestion avoidance* continua até ser detectado congestionamento
 - » Ao aumentar *cwnd*, o TCP está a “testar” a rede (*probing*) até que comecem a ocorrer perdas, altura em que a janela de congestionamento deve ser reduzida

Evolução da janela de congestionamento

- *Slow start* – aumento exponencial



- *Congestion avoidance* – aumento linear



TCP – recuperação após congestionamento

- A ocorrência de *time-out* e a recepção de ACKs duplicados são usados como indicação de perda de pacotes e, portanto, para desencadear acções de recuperação
 - » O receptor TCP deve enviar um ACK (duplicado) quando receber um pacote fora de ordem
- Se ocorrer um *time-out* são feitas as actualizações seguintes (*multiplicative decrease*)
 - » O valor de *ssthresh* é alterado como estabelecido no RFC 2581

$$ssthresh = \max (FlightSize / 2, 2 * SMSS)$$
 - » O valor de *cwnd* é reduzido a um segmento (SMSS)
 - » O emissor é colocado no estado *slow start*
- A recepção de três ACKs duplicados (sem que ocorra *time-out*) é uma indicação forte da ocorrência de perda de um pacote mas também que o receptor continuou a receber novos pacotes – neste caso realiza-se *fast retransmit* e *fast recovery* (e não *slow start*)
 - » Após recepção do terceiro ACK duplicado, *ssthresh* é actualizado com o valor acima referido
 - » O segmento perdido é retransmitido e actualiza-se $cwnd = ssthresh + 3 * SMSS$ (a janela de congestionamento é aumentada artificialmente com os três segmentos que deixaram a rede)
 - » Por cada novo ACK duplicado recebido, *cwnd* é aumentado de um valor igual a SMSS (corresponde ao novo segmento que deixou a rede) e é transmitido um novo segmento, se tal for permitido pelo valor actualizado da janela – $\min (cwnd, rwnd)$
 - » Após recepção de um ACK que efectivamente confirma dados, *cwnd* assume um valor igual ao *ssthresh* actualizado no início da recuperação (reduzindo o aumento artificial da janela)

TCP – slow start e congestion avoidance

